Board Station for New Users Training Series

Module 4: Packaging the Design for LAYOUT

Software Version 8.5_2

Mentor Graphics®

Copyright © 1991-1996 Mentor Graphics Corporation. All rights reserved.
Confidential. May be photocopied by licensed customers of
Mentor Graphics for internal business purposes only.

The software programs described in this document are confidential and proprietary products of Mentor Graphics Corporation (Mentor Graphics) or its licensors. No part of this document may be photocopied, reproduced or translated, or transferred, disclosed or otherwise provided to third parties, without the prior written consent of Mentor Graphics.

The document is for informational and instructional purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in the written contracts between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

A complete list of trademark names appears in a separate “Trademark Information” document.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

This is an unpublished work of Mentor Graphics Corporation.

TABLE OF CONTENTS

Lesson 1

TABLE OF CONTENTS [Continued]

Lesson 1

Packaging the Design (Continued)

Lab 1

TABLE OF CONTENTS [Continued]

Lab 1

Packaging the Design (Continued)

Appendix A

LIST OF FIGURES

LIST OF TABLES

About This Training

Welcome to the *Board Station for New Users Training Series*. For information on the tools you learn to use in this training series, see the "About this Training" section of Module 1: *Introduction to Board Station* of the *Board Station for New User's Training Series*.

Workbook Organization

For an overview of the organization and content of all the modules of the *Board Station for New Users Training Series*, refer to section "Workshop Overview" in *Module 1: Introduction to Board Station.*

Related Documentation

For a complete listing of the manuals that make up the PCB documentation set, refer to section "Guide to the Documentation" in the *PCB Products Overview Manual*. The *PCB Products Overview Manual* describes how each manual can help you in the design process. You can find a listing of all Mentor Graphics manuals in the *Mentor Graphics Technical Publications Overview Manual*. Both of these manuals are available in INFORM.

Documentation Conventions

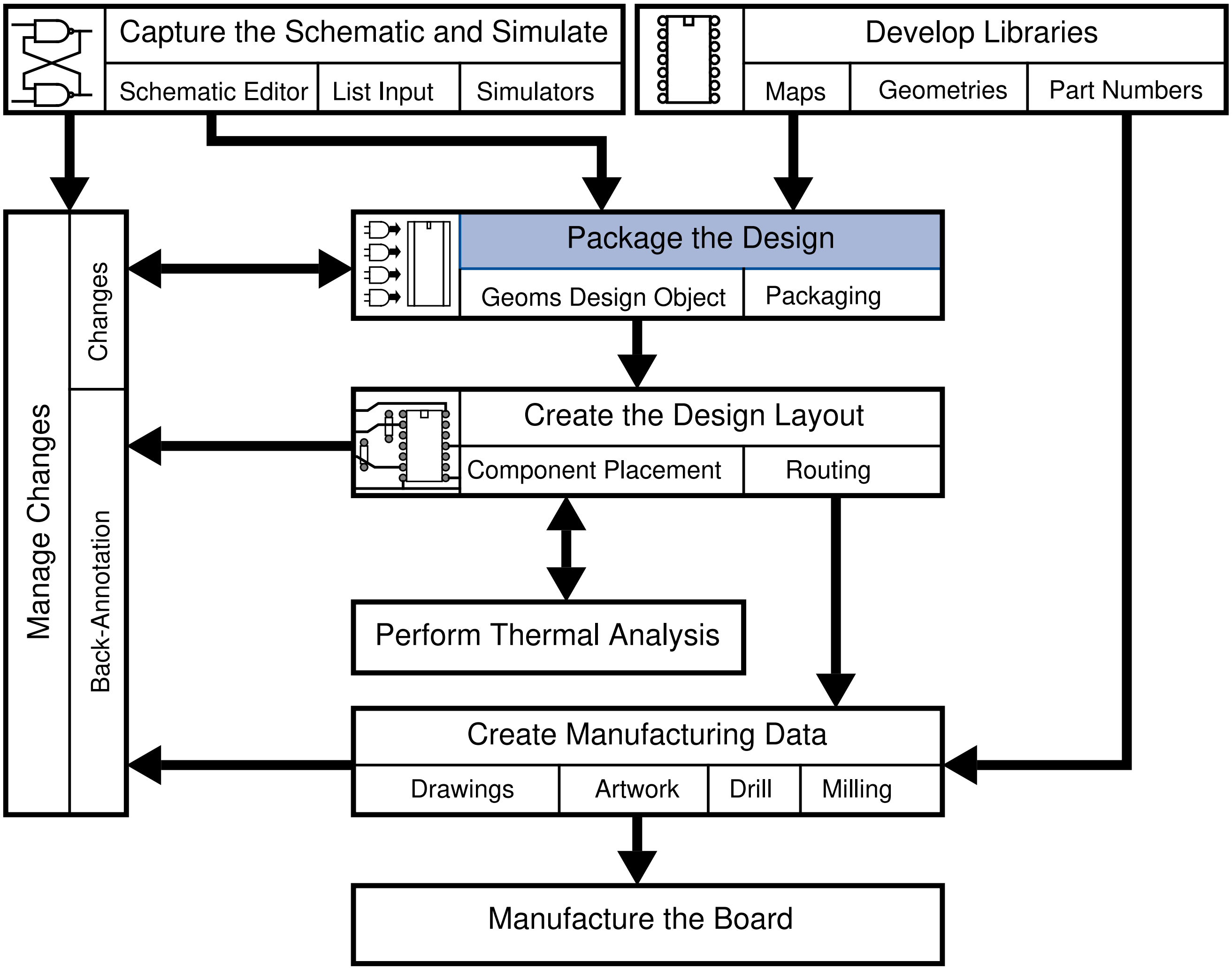
For an explanation of the documentation conventions used in this workbook, refer to the "About this Training" section of Module 1: *Introduction to Board Station* of the *Board Station for New User's Training Series*.

Installation Procedure

For complete instructions on installing the data for this module, refer to "Installation Procedure" in the "About this Training" section of Module 1: *Introduction to Board Station* of the *Board Station for New User's Training Series*.

Lesson 1
Packaging the Design

PACKAGE is the tool that assigns all the symbol instances from the schematic to the specific components needed for the PCB.



Objectives

This lesson explains the concept of Board Station packaging, specifically what you input into the PACKAGE tool. You learn how to use the *pkgconf* design object as a filter for your input, the steps involved in packaging your design, how to create a *geoms* design object. You also learn about the output from a PACKAGE session and how PACKAGE output is used by the LAYOUT tool.

To package a design, you need:

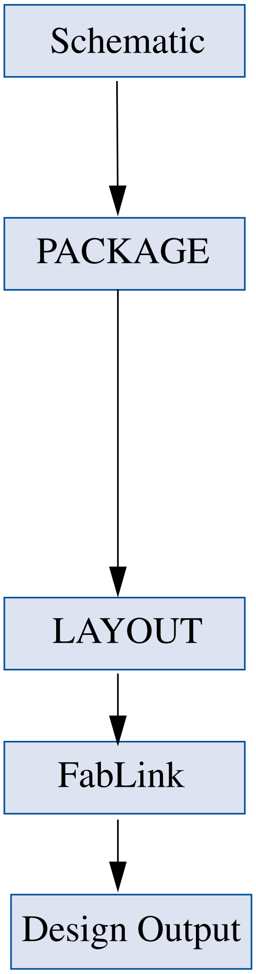
- Geometries
- Catalog files
- Mapping files

for all the symbol types used in the schematic.

You learned to create catalogs and mapping files in the previous module. Often, the catalogs and mapping files you need already exist, so you only need to know the files locations. And, if your company has a Librarian in charge of building PCB parts, including boards and padstacks, the geometries you need for your design already exist. If the catalogs, mapping files, and geometries for your design already exist, packaging the design is where the work of a PCB Design Specialist begins using Board Station. In this case, the PCB Design Specialist never has to use LIBRARIAN.

In some companies, packaging the design is the responsibility of the design engineering group that generated the netlist. In those companies, the work of the PCB Design Specialist begins with the Board Station LAYOUT tool, which is used to place components and route the connections. Even if your responsibilities do not include packaging the design, it is important to understand this stage of the design process.

Where PACKAGE Fits In



PACKAGE is the PCB tool that assigns the logic symbols from a schematic to the physical components, which are then placed on the printed circuit board in LAYOUT. PACKAGE transfers the continuity of the netlist or schematic to the component nets.

PACKAGE creates components by reading either the *comps* and *nets* design objects generated from a nets list and components list, or by reading a schematic created with the Schematic Editor in Design Architect. PACKAGE then assigns logic symbols to geometries.

This lesson assumes the input to PACKAGE is coming from a Design Architect schematic. For information on using a nets list and a components list as the input to PACKAGE, see Appendix A: Entering PACKAGE Without a Schematic.

In LAYOUT, you place components on the printed circuit board and route traces to connect the components. Finally, you create manufacturing output and documentation with FabLink. LIBRARIAN supports the other tools by providing symbol mapping, geometry, and attribute information about the components.

Packaging Concepts

In the schematic, symbol pins provide the input and output to each symbol. Nets provide continuity between the pins. Each net, pin, and symbol has a unique identifier. Figure 1-1 illustrates a small schematic, consisting of AND gates (with an instance name prefix of A) and OR gates (with an instance name prefix of OR). Each symbol has three pins: pin 1, pin 2, and pin 3. The nets are also named.

Ignoring the inputs and outputs, the nets in the above drawing are:

- NET 1 - gate A1, pin 3; gate OR2, pin 1
- NET 2 - gate A2, pin 3; gate OR2, pin 2; gate A3, pin 1; gate A4, pin 2
- NET 3 - gate OR1, pin 3; gate A3, pin 2; gate A4, pin 1; gate OR3, pin 1

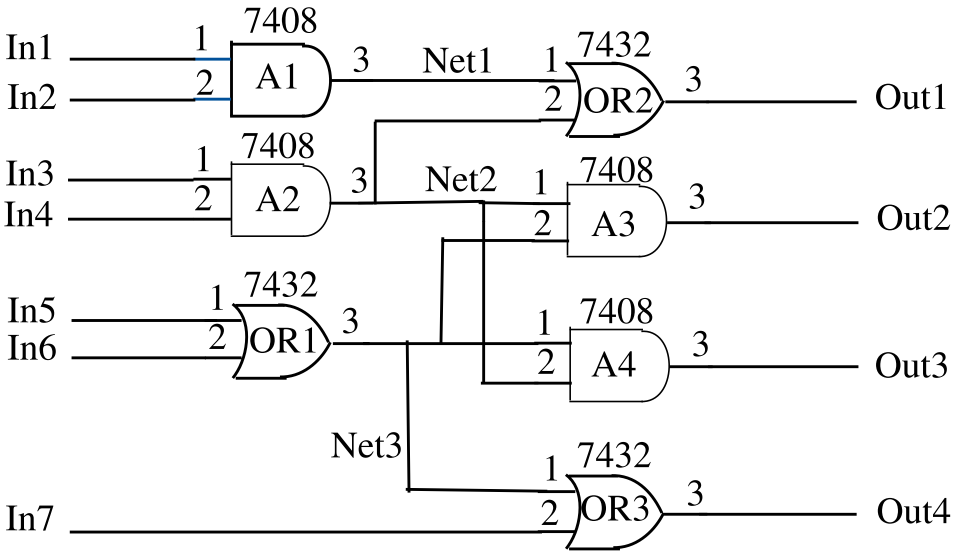


Figure 1-1. Nets Provide Continuity

PACKAGE assigns the symbol instances from the schematic into PCB components. Each symbol pin is mapped to a component pin. On the schematic, the symbol pins are connected to nets; in packaging, the component pins are assigned to the same nets, maintaining connectivity. Figure 1-2 shows how the illustrated schematic can map into components in PACKAGE.

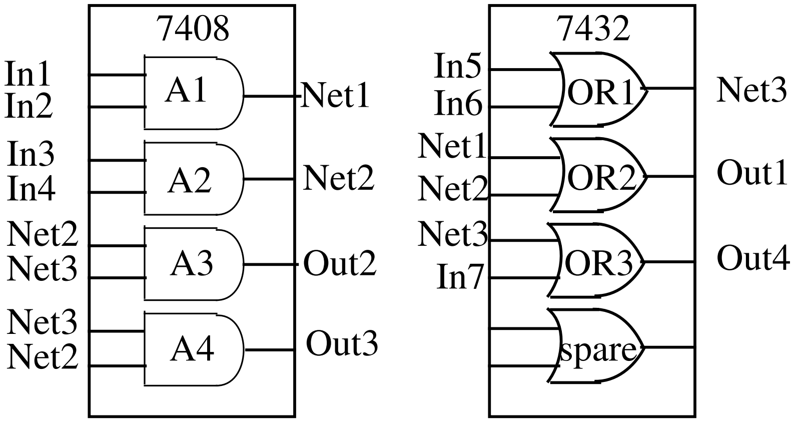
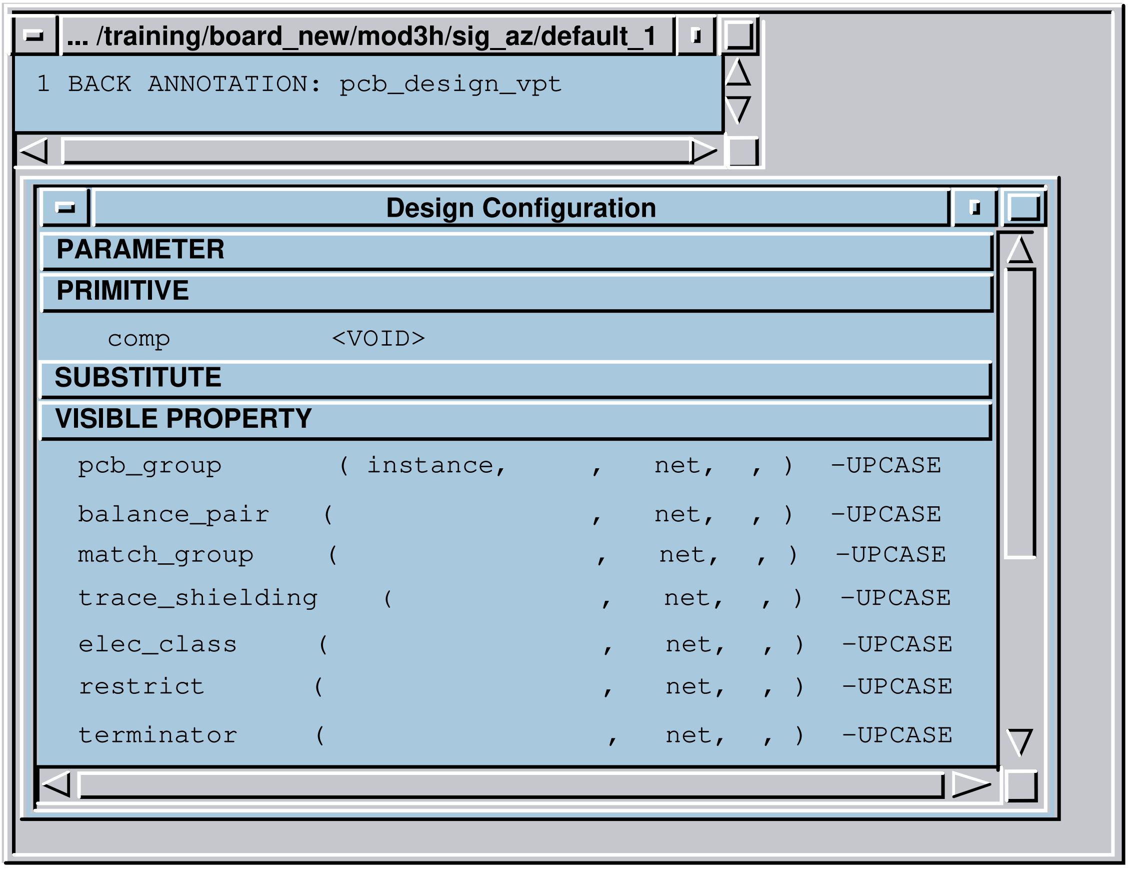


Figure 1-2. Packaging the Schematic

PACKAGE uses the values of the Comp properties to recognize schematic symbols. A schematic symbol can be a resistor, an inverter, an AND gate, a flip-flop, a microprocessor, or any part of a circuit that you designate as a symbol. The Comp property values in the illustrated schematic are 7408 and 7432.

PCB Viewpoint

When you invoke the PACKAGE tool, it creates a PCB viewpoint if one does not already exist. A PCB viewpoint created by the PACKAGE tool is a default PCB viewpoint. The default PCB viewpoint defines the Comp property as the primitive and contains the default PCB visible properties.



... /training/board_new/mod3h/sig_az/default_1

```
1 BACK ANNOTATION: pcb_design_vpt
```

Design Configuration

PARAMETER

PRIMITIVE

```
   comp            <VOID>
```

SUBSTITUTE

VISIBLE PROPERTY

```
  pcb_group        ( instance,      ,    net,   , )   -UPCASE
  balance_pair    (                 ,    net,   , )   -UPCASE
  match_group      (                ,    net,   , )   -UPCASE
  trace_shielding    (              ,    net,   , )   -UPCASE
  elec_class      (                 ,    net,   , )   -UPCASE
  restrict        (                 ,    net,   , )   -UPCASE
  terminator      (                 ,    net,   , )   -UPCASE
```

Figure 1-3. Default PCB Viewpoint

Once PACKAGE creates the viewpoint for the design, you can add, change, or delete properties using the Design Viewpoint Editor, DVE.

For a complete list of the default PCB properties, see the "Properties" section of the *PCB Products Design Reference Manual*.

Input to PACKAGE

In addition to the schematic or netlist, several other files and design objects must be available before you can successfully enter and complete PACKAGE. The files PACKAGE reads include:

- Design Architect schematic or the *comps* and *nets* design objects. The schematic is located in your design container and is read automatically when you enter PACKAGE.

  If the input is from a netlist, rather than a Design Architect schematic, Board Station reads the *comps* and *nets* design objects that were generated from the netlist.

- catalog files. Catalog files contain the part descriptions that guide the symbol-to-component packaging. You must load one or more catalog files after entering PACKAGE and before beginning assignments.

- mapping files. Mapping files associate symbol pins with component pins. Mapping files are made available automatically when their catalog is loaded.

- PACKAGE configuration design object, named *pkgconf*. *Pkgconf* identifies non-standard property names on the schematic and contains information that controls certain PACKAGE operations.

- geometry files. Geometry files contain geometric descriptions of components. You load one or more geometries when you want to check the geometries or create a *geoms* design object for the design in PACKAGE, instead of in LIBRARIAN.

If you saved your design data from a previous PACKAGE session, PACKAGE can read in the *comps*, *gates*, and *nets* design objects from the previous session.

Pkgconf Functions

The Package Configuration *(pkgconf)* design object acts as a filter between the logic information on the net list or schematic and Board Station. *Pkgconf* alerts PACKAGE to look for non-standard property names on the schematic symbols. The standard property names are defined in the "Properties" section of the *PCB Products Design Reference Manual*.

The *pkgconf* design object has four functions:

1. Enables PACKAGE to pick up schematic symbol property values that are attached to properties with non-standard property names.

2. Controls the way PACKAGE assigns reference designators.

3. Enables PACKAGE to forward annotate instance and net properties to LAYOUT.

4. Controls PACKAGE functions, such as the selection of part numbers and the recognition of connectors.

The *pkgconf* design object includes statements that you specify to control the listed actions. These statements are discussed on the following pages.

The *pkgconf* design object must be created before running PACKAGE. *Pkgconf* must be located in the *pcb* container under the design directory, with the rest of the design objects relevant to your design.

You create the *pkgconf* design object in LIBRARIAN by choosing the **Setup > Pkgconf** menu item. A dialog box appears that allows you to customize the various options, which are called statements.

If you enter PACKAGE without a *pkgconf* design object in the design directory, PACKAGE uses a system default file called *pkg.config*. An example *pkg.config* file exists at *$MGC_HOME/pkgs/pcb_base/data/pkg.config*. You need to edit this example *pkg.config* file to include the statements you need, to use it as a system default.

Component / Instance / Net / Pin Statements

The Instance, Net, Component, and Pin statements allow PACKAGE to recognize user-defined instance, net, component, and pin properties names in the schematic.

The formats of the Component, Instance, Net, and Pin statements are:

```
Component  <property_name>
Instance  <property_name>
Net  <property_name>
Pin  <property_name>
```

Equivalent Statement

You can use the Equivalent statement to have symbol values attached to properties that are not standard Board Station properties. With the Equivalent statement, you inform PACKAGE that a non-standard property is the same as a particular Board Station property.

A common reason to use the Equivalent statement is when importing a netlist from another CAD system into Board Station. That system may use an identifier other than the Comp property to identify the symbols.

For example, all of the symbols in your design have the property Name rather than Comp. Enter the following statement in the *pkgconf* design object to inform PACKAGE that the Name property is the same as the Comp property:

```
Equivalent  Comp  Name
```

Now PACKAGE looks for the Name property rather than Comp.

You might have a case where some of the symbols in your design have the Comp property and some of the symbols have the Name property. Use the Equivalent statement in the following form to tell PACKAGE to look first for the Comp property, then if the Comp property is not found, look for the Name property:

```
Equivalent Comp Comp Name
```

If both Comp and Name properties are present for the same symbol instance, the Name property is ignored, since the Comp property comes first in the Equivalent statement.

In packaging non-homogeneous parts, an alternate Comp property is frequently required to differentiate the symbols in the part. In this case, the Equivalent statement directs PACKAGE to look for the alternate property before looking for the Comp property, as follows:

```
Equivalent  Comp  Name  Comp
```

This statement directs PACKAGE to look first for the Name property. If the Name property is not found, PACKAGE looks for the Comp property.

Reference Statement

PACKAGE automatically assigns a reference designator to each component it creates. By default, PACKAGE assigns the reference designator prefix U for all components except connectors; PACKAGE assigns J for recognized connectors.

The Reference statement controls the reference designator prefix used in automatic assignment. The Reference statement designates a reference designator prefix for PACKAGE to use and, optionally, designates that the prefix applies only to components containing specific schematic symbol type(s).

You can change the default with the following kind of Reference statement:

```
Reference  Z
```

Suppose you want the reference designators of resistors to start with R, such as R1, R2, or R3. For this example, assume the Comp property value for resistors is Res.

```
Reference  R  Res
```

You can assign a particular reference designator prefix to more than one type of schematic symbol. Each type of schematic symbol has a unique Comp property value. The following statement assigns the reference designator prefix C to components having either Cap or Ecap as the Comp property value.

```
Reference  C  Cap  Ecap
```

You can also use the question mark (?) as a wild card for trailing text in the Comp property portion of a Reference statement. The following statement assigns a reference designator prefix of R to any symbol whose Comp property value begins with Res.

```
Reference  R  Res?
```

Annotate_portions Statement

You can choose to annotate portion text as part of the reference designator. For example, when U15-A, J2-B53, U20-3 are reference designators, the A, B53, and 3 are the portion text.

The Annotate_portions statement controls this process. Back annotation means passing property information from the board design back to the PCB design viewpoint.

Portion text refers to the name that you assign to logic symbol occurrences within a part number. When you create a part number in LIBRARIAN, and you map a logic symbol to the part number and there are multiple occurrences of a symbol for the part number, you assign a portion text value for each symbol occurrence.

For example, in the part number for a DIP14 you map four logic symbols to the DIP geometry. Each symbol represents one portion within the geometry. Portion is equivalent to symbol count. Each portion has an identifier, usually either a number (1, 2, 3, 4, ...) or a letter (A, B, C, D, ...). When you map the logic symbol to the part number, you are asked to enter the identifier for the first symbol. The system then increments the identifier for each additional symbol in the part number to follow the pattern you have established. The identifier is the portion text.

As part of normal back annotation, the reference designator assigned to a component is passed back to the schematic and appears on each of the symbols on the schematic. For example, if the board has a DIP with the reference designator U14 and the DIP contains four logic symbols (portions 1, 2, 3, 4), back annotation marks four symbols on the schematic with the U14 reference designator.

If you have Annotate_portions turned on, back annotation not only passes back the reference designator, but also the symbol portion text. One symbol on the schematic is marked as U14-1, the next as U14-2, the next as U14-3, and the last as U14-4.

The Annotate_portions statement affects only the back annotation of references from PACKAGE to the PCB design viewpoint. LAYOUT and FabLink annotate references based on the form of the references already existing in the PCB design viewpoint. The value of the Ref property (whether U5 or U5-1) is what gets back annotated. For LAYOUT to back annotate the portion text, Annotate_portions must be turned on when you back annotate from PACKAGE.

When PACKAGE reads a design, it attempts to use forward annotated portion information. The term forward annotated means passing property information from the schematic forward into Board Station. In the case of portion text, if the schematic engineer wants to be certain logic symbols are packaged together, the engineer can assign the reference designator value to the symbols. For example, four symbols are marked with the reference designator of U14. If, as part of the reference designator value, the engineer includes portion text, one symbol is U14-1, the next U14-2, and so on.

PACKAGE can recognize these assignments as portion text when the schematic properties are read, or forward annotated, into PACKAGE. PACKAGE attempts to honor the specific portion assignments during packaging, unless an error would result.

If you enter the statement:

```
Annotate_portions On
```

in the *pkgconf* design object, PACKAGE includes the portion text (-A, -B53, and -3 in the previous examples) with the references when you back annotate.

With the statement:

```
Annotate_portions Off
```

PACKAGE does not include the portion text with the references when you back annotate. Annotate_portions Off is the default condition.

You can also choose to annotate portions only for recognized connectors. The statement to use is:

```
Annotate_portions Conn
```

Connector Statement

PACKAGE keeps track of components designated as connectors. If a symbol type is designated as a connector, the component into which the symbol is packaged is considered a connector in PACKAGE. PACKAGE recognizes connector symbol types if the Comp property values begin with the text **conn** or **edge**.

If the Comp property of a connector symbol contains any other value, inform PACKAGE of the value using the Connector statement.

For example, the Connector statement:

```
Connector  edcon
```

tells PACKAGE that any schematic symbol instance with a Comp property value of **edcon** is a connector. As a result, any **edcon** symbol is considered a connector by PACKAGE.

Alias_power Statement

Mapping files must have power pins defined to match the power nets on the schematic. Because different schematics can use different power net names, mapping files from one design do not necessarily apply to another design without changing the power pin definition or definitions in the mapping file.

To provide an alternate name for the power nets in existing mapping files, enter an Alias_power statement in the *pkgconf* design object. Using an alias for power net names in mapping files makes the mapping file design independent.

For example, your mapping files have power defined as vcc. On the schematic, the designer used +5v as the power. You can use the mapping files with power defined as vcc using the Alias_power statement.

In the Alias_power statement you include the power net name from the mapping file and, optionally, an alternate name for the power net. The power net names are case sensitive.

Alias_power *map_file_pwr_net alt_pwr_net*

Using the example, the Alias_power statement looks as follows:

Alias_power vcc +5v

Units Statement

In a schematic or Comps design object created on a release of Board Station prior to V7.0, you may have component locations in user units, where the unit used is not defined. To use the schematic or *comps* design object you need to tell PACKAGE how to interpret the locations. The Units statement tells PACKAGE how to interpret the x,y locations.

Additionally, you might want to control the user units used to back annotate the Brd_loc property from PACKAGE, without reading in a board geometry. The Units statement defines the user units to enter as part of the Brd_loc property value.

The format of the Units statement is as follows:

```
Units  <unit_of_measure>
```

The unit_of_measure is one of the following:

- inch—indicating user units are in inches.
- cm—indicating user units are in centimeters.
- mm—indicating user units are in millimeters.
- mils—indicating user units are in mils (.001 inch).
- tmil—indicating user units are in tenth-mils (.0001 inch).
- tnano—indicating user units are in ten nanometers (.00005).
- none—indicating there are no user units defined.

Part Number Selection

One of the purposes of the *pkgconf* design object is to control how PACKAGE matches a symbol instance from the schematic with a part number entry in a catalog file.

When selecting part numbers, PACKAGE looks at the Mentor Graphics catalogs and at catalogs you load into the PACKAGE session. PACKAGE selects the first part number it encounters that meets the criteria set for packaging. Three *pkgconf* statements affect the criteria for part number selection:

- Default_catalogs statement
- Assign_prop statement
- Select_pn_by_prop statement

Default_catalogs Statement

PACKAGE uses the pathname of the symbol to determine the Mentor Graphics catalog to read. If the pathname of the symbols in your design is the same as the Mentor Graphics library hierarchy, PACKAGE attempts to read the Mentor Graphics catalogs by default to find a part number for a symbol.

For example, a symbol from the symbol library
$MGC_PCBPARTS/pcb_libs/ls_lib

results in an attempt to read the Mentor Graphic catalog
$MGC_PCBPARTS/pcb_maps/ls_maps/ls.catalog.

If PACKAGE finds a match, PACKAGE uses the Mentor Graphics part number rather than a part number in a user catalog.

You can control this process with the Default_catalogs statement.

```
Default_catalogs  On|off
```

The default condition for this process is On.

By setting Default_catalogs Off, PACKAGE does not read the Mentor Graphics first when looking for a part number.

Assign_prop Statement

You can specify the properties for PACKAGE to use in selecting the part number for a symbol. By default, Build selects a part number if the symbol contains a subset of the properties found in the catalog entry. The Assign_prop statement allows you to define the properties that must match between the symbol and the part number.

For example, the properties Value and Rating might be key to selecting the correct part number for any symbol with a Comp property of Res. The following statement ensures that Build selects a part number where the values of the Value property and the Rating property are the same as the values of those properties on the symbol.

```
Assign_prop  Res  value  rating
```

During the next Build, PACKAGE compares the values of the Value and Rating properties on the Res symbols against the catalog entries. PACKAGE selects only a part number with property values that match the symbol property values exactly.

If PACKAGE cannot find a match, PACKAGE issues an error message and halts the Build operation. You can then read in additional catalogs, create a new part number description, or modify the property information on the symbol instances to resolve the error condition.

Select_pn_by_prop Statement

Select_pn_by_prop is similar to the Assign_prop statement. Select_pn_by_prop is more global, however.

During automatic assignment, PACKAGE selects a part description from the loaded catalog files by matching the Comp property of the symbol with an entry in a catalog file.

Recall that the Assign_prop statement allows you to specify properties on the symbol that must match properties in the part number description in the catalog file for that part number to be selected. Setting the Select_pn_by_prop statement to On specifies that all properties on the symbol must match properties in the part number description in the catalog file for that part number to be selected.

If multiple part number descriptions match the Comp property of the symbol and Select_pn_by_prop is set to Off, PACKAGE selects the first part number is encounters that matches the Comp property of the symbol.

The following statement:

```
Select_pn_by_prop  On
```

directs PACKAGE to look for a match between the all the property values on the symbols and the property values defined in the part number description in the catalog file.

Ignore_props Statement

If PACKAGE recognizes a property on the schematic or in a catalog file, PACKAGE reads the property into the packaging session and back annotates the property to the PCB design viewpoint. You can control the properties included in the packaging session using the Ignore_props statement. The Ignore_props statement directs PACKAGE to ignore the specified properties coming from the schematic or the catalog file.

The Ignore_props statement allows you to have one catalog containing all the properties for your design and analysis tools. For example, one catalog could contain all the properties for the Board Station tools and for AutoTherm. By selectively ignoring unneeded properties, you can ensure that PACKAGE reads into the session only the properties you need. Build does not consider any ignored properties. Similarly, back annotation does not back annotate any ignored properties to the PCB design viewpoint.

```
Ignore_props  <property_type>  [property_name]
```

The *pkgconf* design object accepts three types of Ignore props statements to control the flow of properties. The property types are:

- Schematic—instructs PACKAGE to ignore the specified property or properties on the schematic, even if the property is defined as visible in the PCB design viewpoint.
- Catalog—instructs PACKAGE to ignore the specified property or properties coming from the catalog.
- Annotate—instructs PACKAGE not to back annotate the specified property or properties to the PCB design viewpoint.

This example directs PACKAGE to ignore the properties junction_t_max and pow_del_max that exist in a catalog.

```
Ignore_prop  Catalog  junction_t_max  pow_del_max
```

Dump_symbol_map Statement

PACKAGE can create a mapping file from the information contained on a symbol rather than being created in LIBRARIAN. Building a mapping file from symbol information requires that all the physical information exist on the symbol.

To direct PACKAGE to get the mapping information from the symbol, enter the string *nomap* in the mapping file field of the part number entry in the catalog file. During a Build, PACKAGE then reads the mapping information from the symbol.

By default, PACKAGE reads the mapping information from the symbol and uses it to package the symbol. PACKAGE then discards the mapping information. The Dump_symbol_map statement directs PACKAGE to write the mapping information to a mapping file instead of discarding it. The name of the newly created mapping file is <symbol_name>.map.

Adding a directory name as the argument to the Dump_symbol_map statement specifies the directory location for the mapping file. Without the directory_name argument, PACKAGE writes the mapping file to the design directory.

The format of this optional statement is:

```
Dump_symbol_map  [directory_name]
```

Disconnected_common_allowed

Instances of the same symbol type may not be packaged together if one instance has a disconnected common pin and the other instance has a connected common pin. The disconnected common pin on the first instance becomes shorted to the connected common pin on the second instance, violating schematic connectivity. The symbol instances must be packaged separately.

The exception is where all instances of the same symbol type have disconnected common pins. PACKAGE does allow instances of the same symbol type, neither of which have connected common pins, to coexist in the same package.

You can override this default behavior using the Disconnected_common_allowed statement. The Disconnected_common_allowed keyword allows gates with disconnected common pins to be packaged with gates whose common pins are attached to a single net.

Refer to Figure 1-4 on the next page. The common pins on three of the gates are hooked to a single net on the schematic. The common pin of the fourth gate is a dangling pin. Design Architect assigns a name to the net to which the fourth gate is attached.

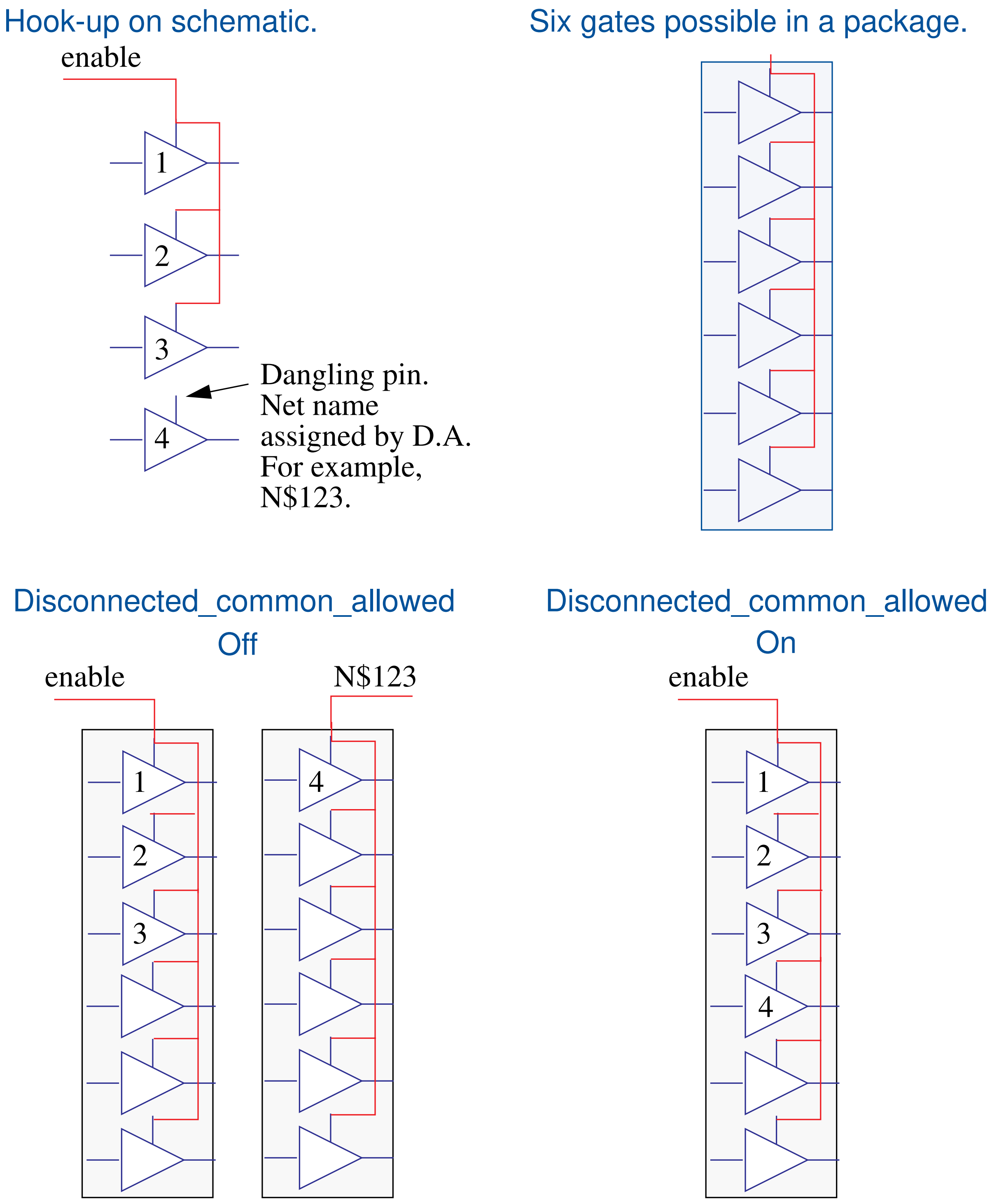
If you set Disconnected_common_allowed to Off, Build packages the fourth gate separately from the other three. If you set Disconnected_common_allowed to On, all four gates are packaged together.

Setting Disconnected_common_allowed to On actually changes the connectivity of the fourth gate. The designer must determine whether this changes the function of the circuit and whether to set Disconnected_common_allowed On or Off.

The syntax of the Disconnected_common_allowed statement is as follows:

```
disconnected_common_allowed  on|off
```

The default value of Disconnected_common_allowed is On.



Hook-up on schematic.

Six gates possible in a package.

Dangling pin. Net name assigned by D.A. For example, N$123.

Disconnected_common_allowed Off

Disconnected_common_allowed On

Figure 1-4. Disconnected_common_allowed

Write_all_pins Statement

PACKAGE writes pin property information to the *pins* design object. By default, PACKAGE only writes the information from pins containing pin properties. The Write_all_pins statement directs PACKAGE to write information from all pins, regardless of whether they have pin properties attached.

Set the value of the statement to On to write information from all the pins. Set the value to Off to write information only from pins containing pin properties.

The format of this optional statement is:

```
Write_all_pins  on|Off
```

The default condition for this option is Off.

Write_one_pin_nets Statement

You can specify whether PACKAGE writes one-pin nets to the *nets* design object. One-pin nets (wires) are those nets connected to only one pin. One-pin nets can be useful for checking or as reference data; however, LAYOUT ignores one-pin nets. The statement that controls this option is Write_one_pin_nets.

The format of this optional statement is:

```
Write_one_pin_nets  on|Off
```

The default condition for this option is Off.

User-Defined Properties

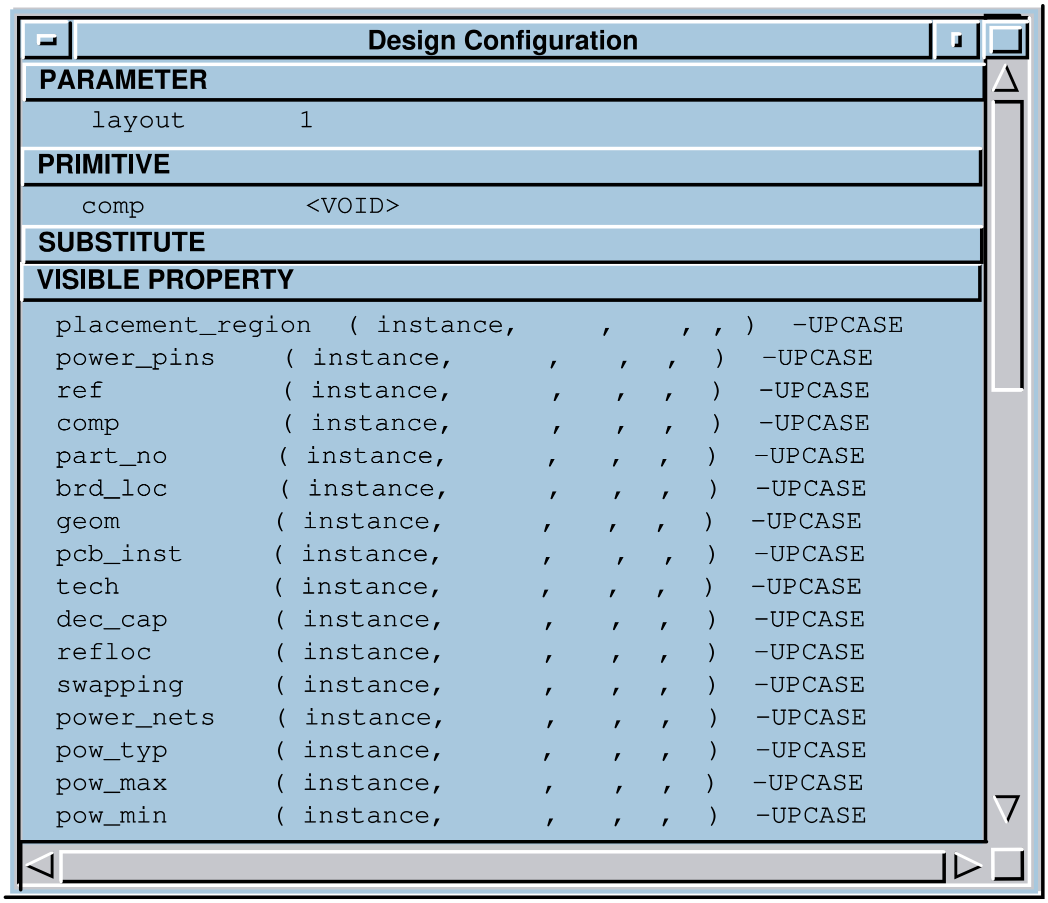
One of the purposes of the *pkgconf* design object is to act as a filter when forward annotating properties to the PACKAGE tool. When forward annotating custom properties defined by you, the *pkgconf* design object works in conjunction with the visible property list in the PCB design viewpoint.

You need to understand the following when defining your own properties:

- Setting up default Board Station properties.
- Defining user-defined properties.
- Recognizing user-defined properties in PACKAGE.
- Using the Design_prop statement.

Default Board Station Properties

You set up the default list of Board Station properties in DVE when you create a PCB Design Viewpoint. If no PCB Design Viewpoint exists when you invoke the PACKAGE tool, PACKAGE creates a default viewpoint and includes the default list of Board Station properties.



```
PARAMETER
    layout        1
PRIMITIVE
   comp              <VOID>
SUBSTITUTE
VISIBLE PROPERTY
  placement_region  ( instance,     ,    , , )  -UPCASE
  power_pins      ( instance,      ,   ,  ,  )  -UPCASE
  ref             ( instance,      ,   ,  ,  )  -UPCASE
  comp            ( instance,      ,   ,  ,  )  -UPCASE
  part_no         ( instance,      ,   ,  ,  )  -UPCASE
  brd_loc         ( instance,      ,   ,  ,  )  -UPCASE
  geom            ( instance,      ,   ,  ,  )  -UPCASE
  pcb_inst        ( instance,      ,   ,  ,  )  -UPCASE
  tech            ( instance,      ,   ,  ,  )  -UPCASE
  dec_cap         ( instance,      ,   ,  ,  )  -UPCASE
  refloc          ( instance,      ,   ,  ,  )  -UPCASE
  swapping        ( instance,      ,   ,  ,  )  -UPCASE
  power_nets      ( instance,      ,   ,  ,  )  -UPCASE
  pow_typ         ( instance,      ,   ,  ,  )  -UPCASE
  pow_max         ( instance,      ,   ,  ,  )  -UPCASE
  pow_min         ( instance,      ,   ,  ,  )  -UPCASE
```

Figure 1-5. PCB Viewpoint in DVE

Defining User-Defined Properties

In addition to the default Board Station properties, you can define your own property and attach that property to an instance on the schematic.

For Board Station to recognize a user-defined property, you must add the new property in two places:

- The visible property list in the PCB Design Viewpoint.

  You add the property to the PCB Design Viewpoint using the Design Viewpoint Editor, DVE.

- The *pkgconf* design object.

  You create the *pkgconf* design object in LIBRARIAN by choosing the **Setup > Pkgconf** menu item.

If a PCB Design Viewpoint does not exist when you invoke the PACKAGE tool, PACKAGE creates a default design viewpoint for you.

However, if you want to use a user-defined property, you must create your PCB design viewpoint and add the user-defined property to the viewpoint before you invoke the PACKAGE tool. You must also define the property in the *pkgconf* design object using the Component, Instance, Net, or Pin statement.

Recognizing User-Defined Properties

When PACKAGE sees a user-defined property coming from the schematic, PACKAGE looks at both the visible property list in the PCB Design Viewpoint and at the *pkgconf* design object to determine whether to recognize the property and read the property into the PACKAGE session.

A user-defined property must be in both the PCB Design Viewpoint and in the *pkgconf* design object for PACKAGE to recognize the property.

Table 1-1. Recognizing User-Defined Properties in PACKAGE

User-Defined Property	In PCB Design Viewpoint	In Pkgconf	Recognize?
Property_name_A	Yes	No	No
Property_name_B	No	Yes	No
Property_name_C	Yes	Yes	Yes

Design_prop Statement

If PACKAGE does not recognize a property coming from the schematic, PACKAGE does not read the property into the session. To make a property recognizable to PACKAGE, you must add the property to the visible property list in the PCB Design Viewpoint. You must also add the property to the *pkgconf* design object.

The Design_prop statement removes the requirement of adding a user-defined property to the *pkgconf* design object. Setting Design_prop to On directs PACKAGE to read into the session any property coming from the schematic that is defined in the viewpoint, regardless of whether the property is included in the *pkgconf* design object.

The format of the Design_prop statement is:

```
Design_prop  on|Off
```

The default condition for this option is off.

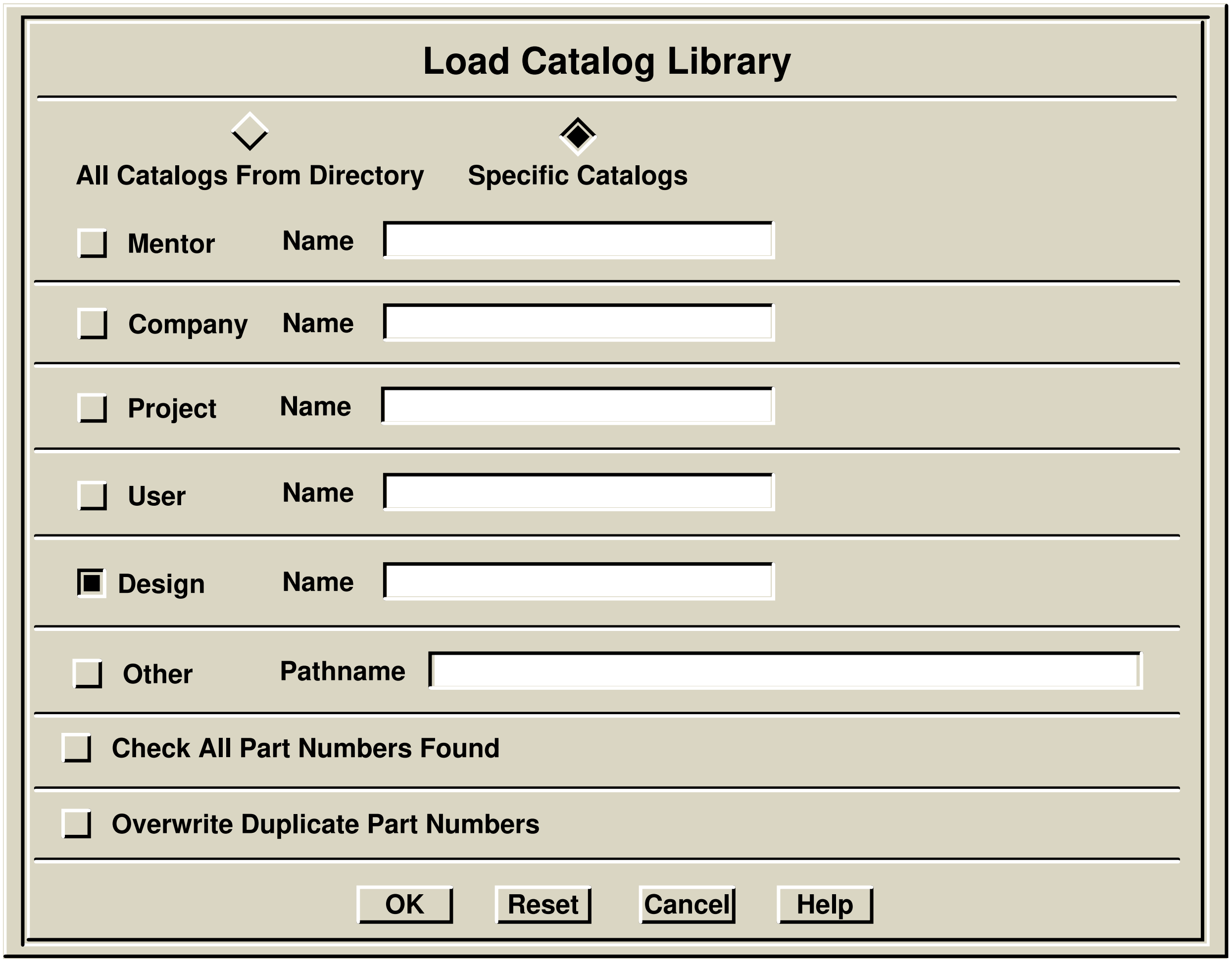
Packaging Process

There are four basic steps to the packaging process.

1. Invoke the PACKAGE tool.
2. Load the needed catalogs.
3. Perform any interactive packaging.
4. Perform a Build to make the part number assignments.

Loading Catalog Files

Before you package the symbols you must load the necessary catalog files. Part number descriptions in the catalog files determine how PACKAGE assigns the schematic symbols into components. Select **Setup > Catalog Library > Load Catalog Library...**



Load Catalog Library

◇ All Catalogs From Directory ◆ Specific Catalogs

□ Mentor Name

□ Company Name

□ Project Name

□ User Name

■ Design Name

□ Other Pathname

□ Check All Part Numbers Found

□ Overwrite Duplicate Part Numbers

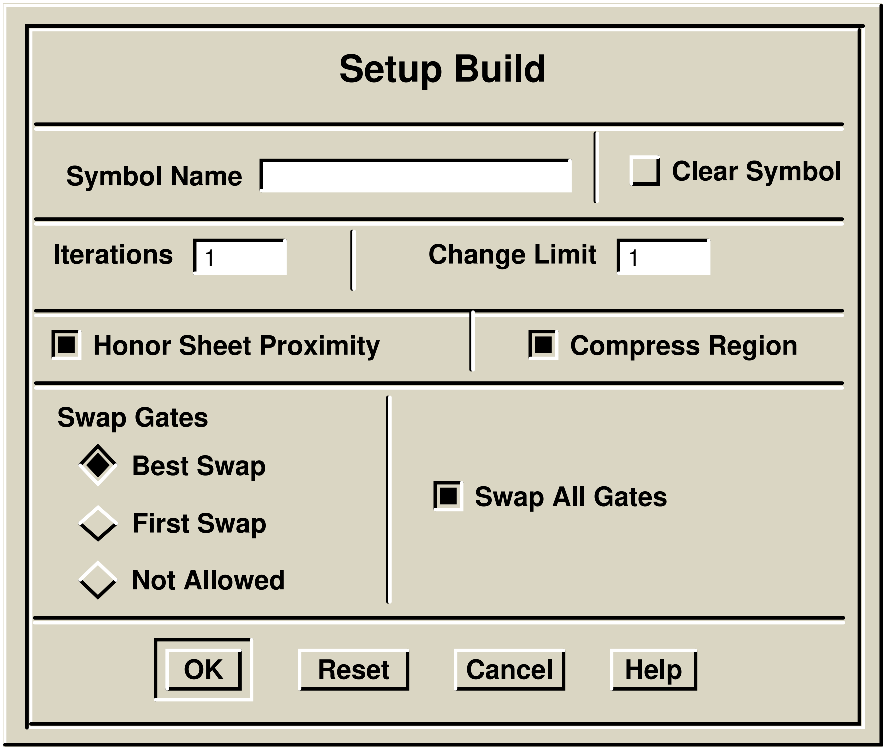
Figure 1-6. Load Catalog Library Dialog Box

Indicate the level of the catalog you are loading, for example, from the Design level. The name *design.catalog* is the default in the dialog box. You can also specify other catalogs, choose and identify them.

You only need to load the catalog data the first time you bring your design into PACKAGE. When the data is saved, the pathnames of catalogs loaded into the PACKAGE design are written to the *catalogs* design object. When you invoke PACKAGE later, the *catalogs* design object is read and the catalogs are automatically loaded.

Setting Build Parameters

You can set up Build parameters for an individual symbol. You can also set Build parameters for all the symbols for which no individual Build parameters exist.



Setup Build

Symbol Name 　 □ Clear Symbol

Iterations 1 　 Change Limit 1

☒ Honor Sheet Proximity 　 ☒ Compress Region

Swap Gates

◆ Best Swap

◇ First Swap

◇ Not Allowed

☒ Swap All Gates

Figure 1-7. Setup Build Dialog Box

The parameters you can set are as follows:

- **Symbol Name**—specifies the symbol type for which you are setting parameters.
- **Clear Symbol**—resets parameters to the default settings.
- **Iterations**—specifies the number of iterations you want Build to perform. In each iteration, Build performs a type of preassignment gate swapping. Build examines each unassigned instance and determines whether to place the instance in a particular component or to keep looking for a better spot for the instance. The goal is to come up with a better build. Enter a positive integer between 1 and 255. The default is 1.

- **Change Limit**—specifies the limit of iterations that result in *no change*. When Build reaches this limit, further iterations are cancelled.

- **Honor Sheet Proximity**—directs Build to attempt to package together gates that are unconnected, but placed close together on the schematic. This parameter does not override gate connectivity.

- **Compress Regions**—directs Build to package the gates into the minimum number of packages, even if this requires assigning gates from more than one circuit group to a single package.

- **Swap Gates**

  Best Swap—directs Build to evaluate all possible swaps and choose the best swap.

  First Swap—directs Build to swap only until the first good swap is found.

  Not allowed—directs Build to assign the next available gate without performing gate swapping.

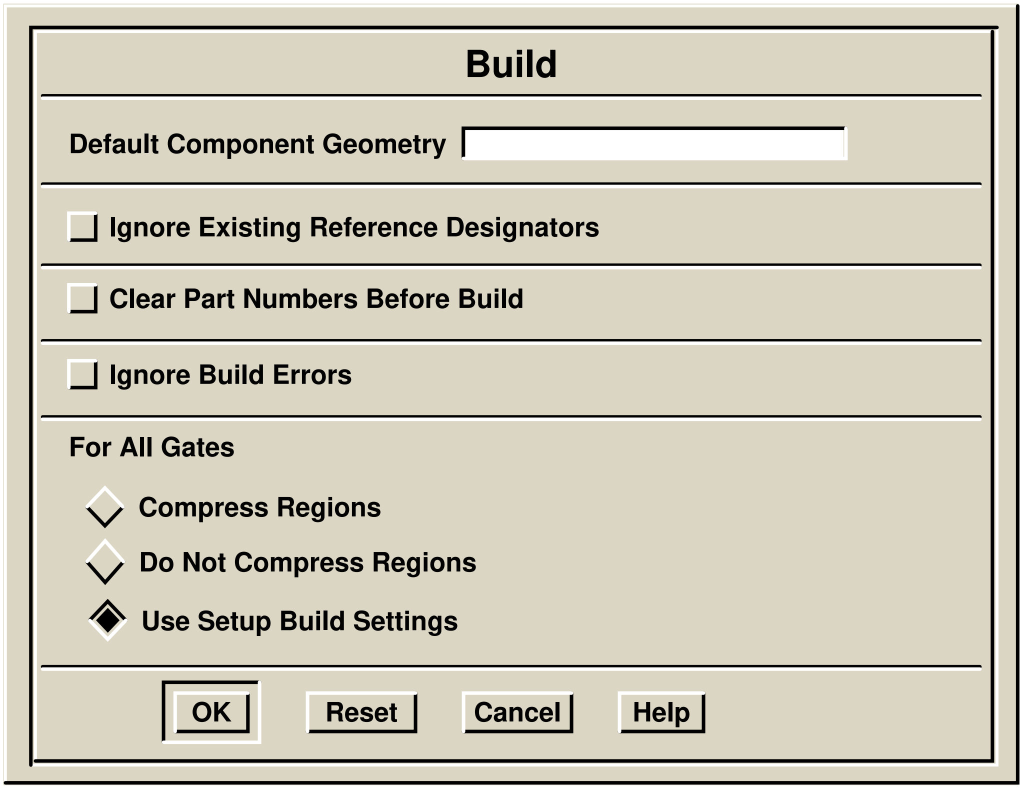
- **Swap All Gates**—enables the evaluation of all gates for possible swapping. Not selecting **Swap All Gates** restricts the gates available for swapping to other unchecked gates only.

Selecting menu item **Report > Build Setup** reports the Build parameters set for symbols in the design.

Automatic Packaging

The Build function automatically assigns all unpackaged symbols into components. The following steps explain what Build does and how to fill out the Build dialog box.

You can select **Build...** from either the Component Summary window or the Edit Symbol window.



Build

Default Component Geometry

□ Ignore Existing Reference Designators

□ Clear Part Numbers Before Build

□ Ignore Build Errors

For All Gates

◇ Compress Regions

◇ Do Not Compress Regions

◆ Use Setup Build Settings

Figure 1-8. Build Dialog Box

The following explains how to fill out the fields in the dialog box.

- **Default Component Geometry**—specifies the geometry to select when a Comp property has multiple geometries.
- **Ignore Existing Reference Designators**—unassigns all unprotected gates before executing Build.
- **Clear Part Numbers Before Build**—clears part numbers that were forward annotated from the schematic or assigned by a previous Build.
- **Ignore Build Errors**—causes Build to ignore most errors so that most assignments are made.

Compress Regions

The group of buttons labeled **For All Gates** affects how Build packages the gates. Each gate can have an associated Placement_region property. This property allows PACKAGE to maintain specific relationships between gates during the Build.

To add properties to gates in PACKAGE select **Properties > Add Property to Gate** from the menu bar and complete the dialog box.

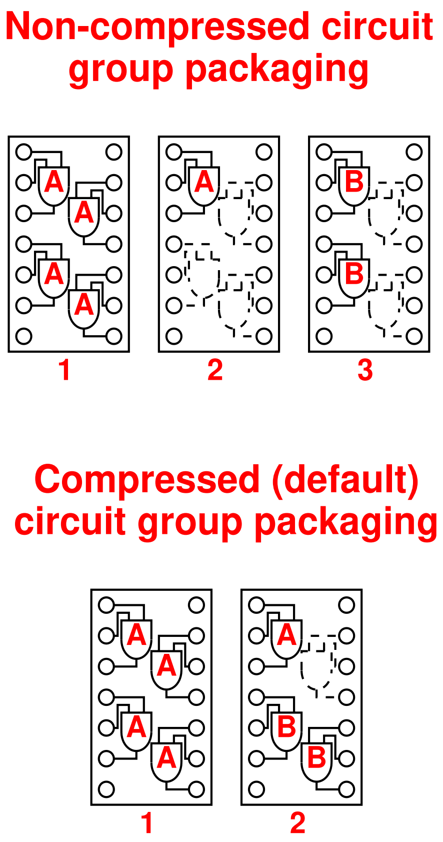

Non-compressed circuit group packaging
A
A
A
A
A
B
B
1
2
3
Compressed (default) circuit group packaging
A
A
A
A
A
B
B
1
2

Figure 1-9. Circuit Group Packaging

- **Compress Regions**—directs Build to package the gates into the minimum number of packages, even if this requires assigning gates from more than one circuit group to a single package.

- **Do Not Compress Regions**—directs Build to not package gates from more than one circuit group in a single package, even though this may result in additional packages being generated.
- **Use Setup Build Settings**—directs Build to respect the compress region settings defined on a per symbol basis using **Setup > Build**.

When assignment to the PACKAGE design takes place, the assignment is indicated by an A flag in the Flag field of the symbol. An *E* flag indicates that Build detected an error. In addition to assigning symbol instances to packages, Build also carries out the key checking function that ensures a consistent design.

For more information, refer to section "Properties" in the *PCB Products Design Reference Manual.*

Interactive Packaging

You can use two methods to package symbols into components: interactive assignment and automatic assignment. With interactive assignment, you can specify the following information about a schematic symbol:

- Geometry
- Location on the PC board
- Mapping file
- Part number
- Reference designator

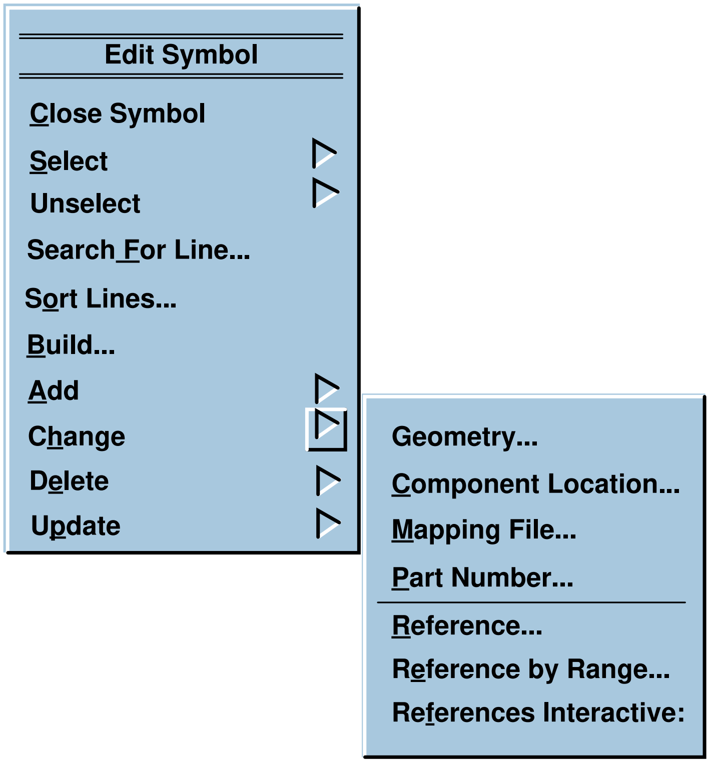


Figure 1-10. Change Menu in the Edit Symbol Window

The results of interactive assignment commands are immediately visible on the screen, but no changes occur in the PACKAGE design until you run Build. When you issue Build for automatic assignment, previous interactive assignments are checked and, if the assignments are correct, they are entered into the PACKAGE design.

You must run Build at the very end of the PACKAGE procedure to check the final interactive assignments and add them to the PACKAGE design.

By the time you have finished the PACKAGE design, each symbol instance is assigned to a component. Assignment progress is recorded in the Component Summary window and the Edit Symbol window. The Component Summary window summarizes the status of the entire design; the Edit Symbol window shows the status of a selected symbol.

Geoms Design Object

If all the necessary geometries have been previously created, you can read them into the PACKAGE session to create a *geoms* design object. You can create the *geoms* design object in LIBRARIAN, rather than in PACKAGE, if you wish. The *geoms* design object must exist before you can bring your design into the LAYOUT tool.

Two prerequisites must be met to enable PACKAGE to check that needed geometries for your design exist in the session.

5. Load the catalog files for your design.

6. Do a Build. Build creates the *comps* design object that PACKAGE uses to determine the geometries needed for your design.

   Check for geometries needed by the design and read them into the session.

Read the geometries required for the design into the PACKAGE session by choosing the **Geometries > Resolve Geometries** menu item.

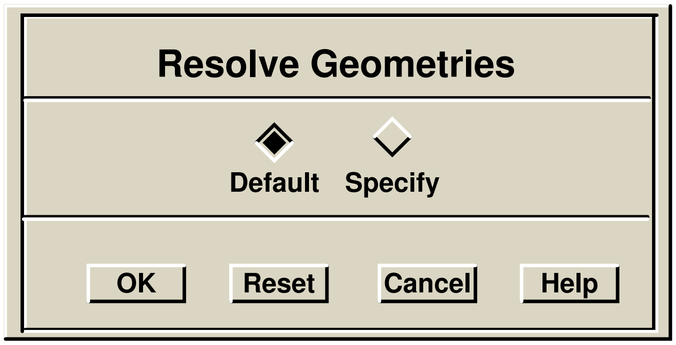


Figure 1-11. Resolve Geometries Dialog Box

The Resolve Geometries dialog box displays.

In the dialog box, you choose to read geometries from either default directories or directories you specify.

If you choose Default, PACKAGE looks for the needed geometries in the *design_geoms* directory, the *pcb_parts* directory under your user directory, and from the Mentor level of the default directory hierarchy. If you choose Specify, you enter the directory name or directory names where PACKAGE searches for any missing geometries.

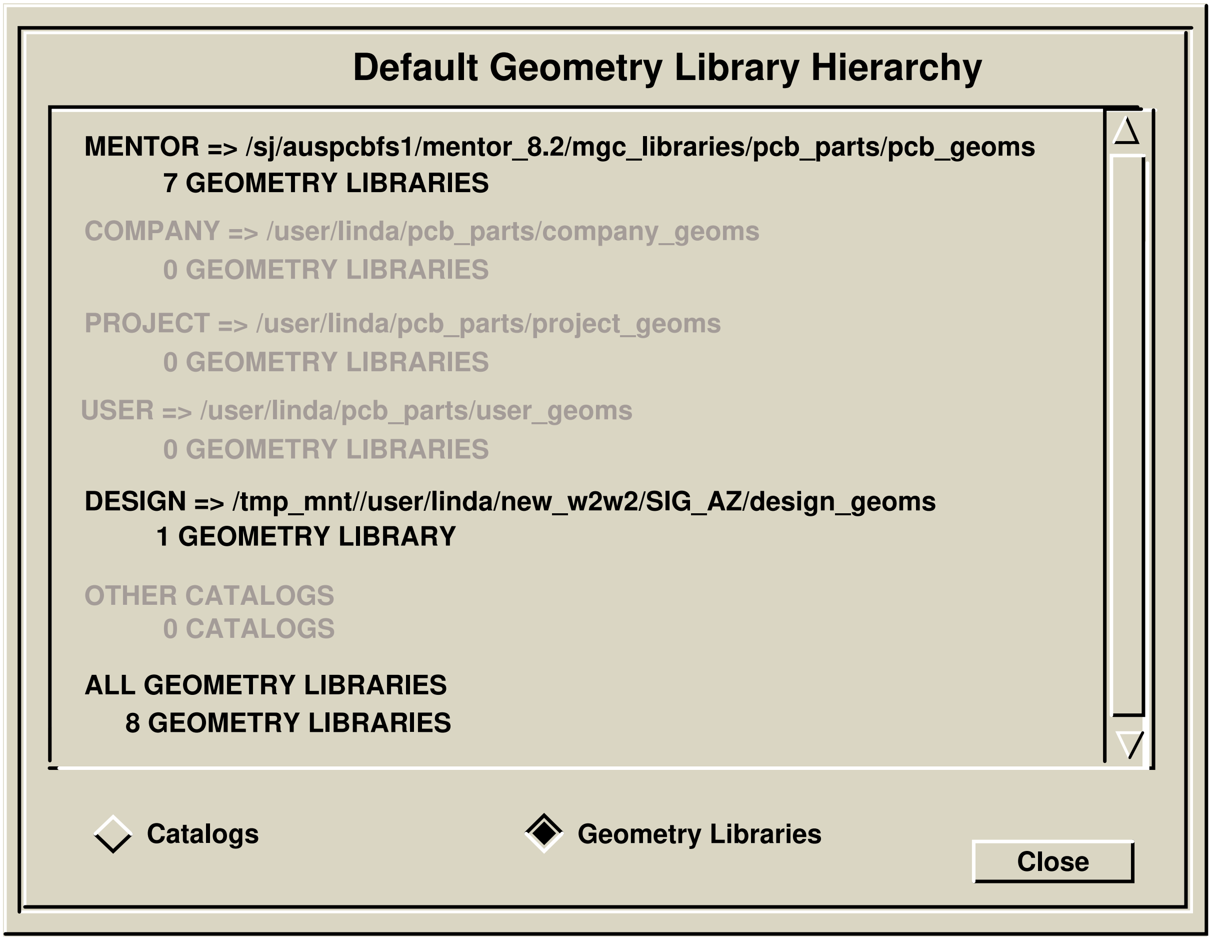
When PACKAGE data is saved, the pathnames of directories loaded into the PACKAGE design are written to the *catalogs* design object.

When you execute the dialog box. Resolve Geometries reports in a Notepad window any geometries that are still needed.

Reading Board, Vias, and Artwork Order

Resolve Geometries does not look for geometries for the board, via, or artwork order geometries; you must read in the board, via, and artwork order geometries.

Read geometries such as the board and vias into the PACKAGE session by choosing the **Geometries > List Geometry Libraries** menu item.



Default Geometry Library Hierarchy

MENTOR => /sj/auspcbfs1/mentor_8.2/mgc_libraries/pcb_parts/pcb_geoms
7 GEOMETRY LIBRARIES

COMPANY => /user/linda/pcb_parts/company_geoms
0 GEOMETRY LIBRARIES

PROJECT => /user/linda/pcb_parts/project_geoms
0 GEOMETRY LIBRARIES

USER => /user/linda/pcb_parts/user_geoms
0 GEOMETRY LIBRARIES

DESIGN => /tmp_mnt//user/linda/new_w2w2/SIG_AZ/design_geoms
1 GEOMETRY LIBRARY

OTHER CATALOGS
0 CATALOGS

ALL GEOMETRY LIBRARIES
8 GEOMETRY LIBRARIES

◇ Catalogs ◆ Geometry Libraries

Figure 1-12. Default Geometry Hierarchy

A dialog box appears displaying the Default Geometry Library Hierarchy.

1. In the dialog box, choose the category of library from which to read the board, vias, or artwork order geometries by clicking the Select mouse button.

   A dialog box appears. The dialog box displays the Geometry Libraries available in the selected category.

2. **C**lick the Select mouse button on a library entry and press the View button in the dialog box.

   The dialog box displays a list of geometries available in the selected library.

3. To read one or more geometries, click the Select mouse button on each geometry, then press the Read button in the dialog box.

   You repeat this process for the board, vias, and artwork order geometries that are required for your design.

Setting up your board to use split power planes is a common reason for an artwork order to exist at this point in the design process.

Back Annotation

In the PCB design process, PACKAGE is the first tool in which you can add or alter property information. One of the tasks of the Build function is to add the Ref property indicating the reference designator when each symbol is assigned. You can also add a new user-defined property in PACKAGE, if the property name exists in the *pkgconf* design object when you invoke the PACKAGE tool.

At any time during a PACKAGE session, you can back annotate property information to the schematic by selecting the menu item **File > Back Annotate**. At the end of a PACKAGE session, when you save the design and close the PACKAGE tool, PACKAGE automatically back annotates property information to the schematic.

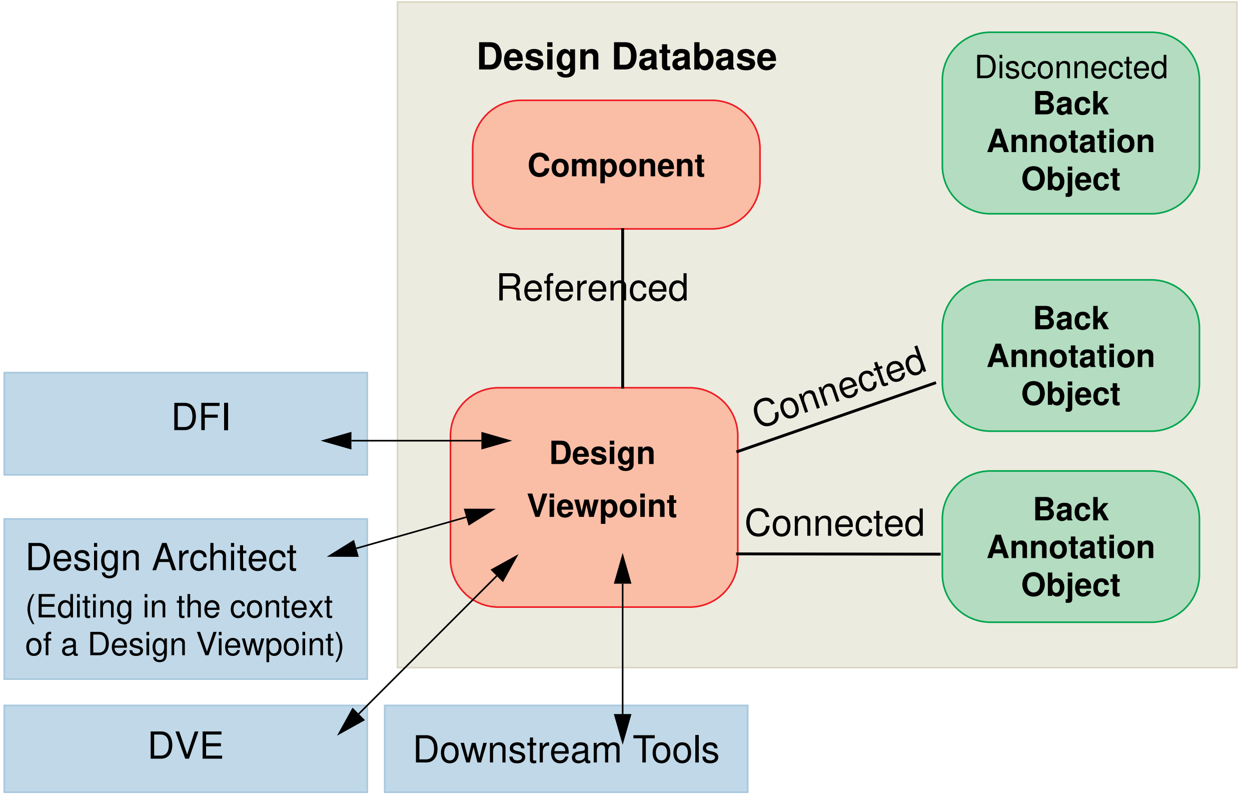
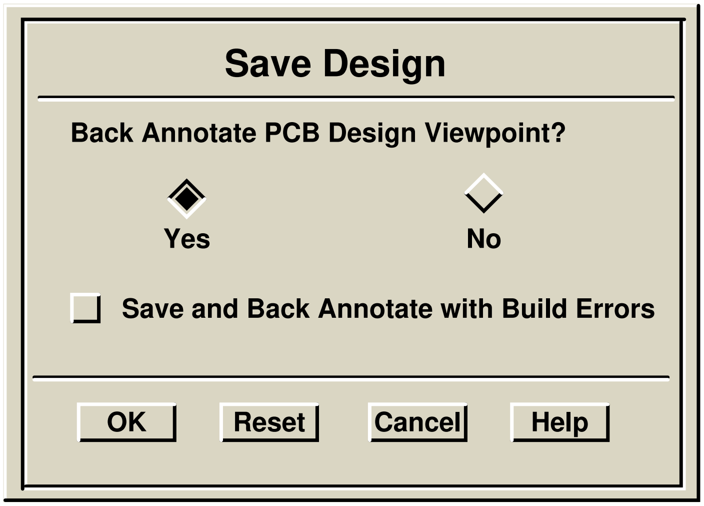


Figure 1-13. Connections Between the Design and Design Objects

Saving PACKAGE Data

You save the data generated during the PACKAGE session by choosing the **File > Save > Design All...** menu item.



Save Design

Back Annotate PCB Design Viewpoint?

◆ Yes ◇ No

□ Save and Back Annotate with Build Errors

Figure 1-14. Save Design Dialog Box

In the Save Design dialog box, you have the option of Back Annotating the PCB design Viewpoint and Saving With Build Errors.

It is good design practice to Back Annotate to the PCB Design Viewpoint. PACKAGE looks for a PCB Design Viewpoint when you invoke. If you leave the PACKAGE tool and come back into PACKAGE later, you must have a back annotated viewpoint for your design to reflect your work from the previous PACKAGE session.

You might choose to save with Build errors in a case where you have done some interactive packaging that you want to save, but your Build has errors and you don’t have time to fix them. Saving with Build errors allows you to preserve the interactive work and go back into PACKAGE later to fix the errors.

PACKAGE saves the following:

- The seven PACKAGE output design objects and back annotates the PCB design viewpoint. The PACKAGE output design objects are the: *comps, nets, gates, pkginst, spares, pkgs,* and *notes.*

- The *catalogs* design object listing the pathnames of the catalogs containing the part numbers for the design. Once saved, PACKAGE reads the *catalogs* design object when it is subsequently invokes. The catalogs are automatically read into the PACKAGE session.

- The geometry descriptions to the *geoms* design object in the design directory. The *geoms* design object is written only if new geometries are read during the PACKAGE session.

You can also save the settings that determine column width and line number display in both the Component Summary window and the Edit Symbol window by choosing the **File > Save Environment** menu item.

PACKAGE Output

To effectively preserve the effects of a PACKAGE session, you must write the design objects that become the input to LAYOUT:

- *comps*—a record of all the components in the design. *Comps* data written from PACKAGE includes reference designators, part numbers, symbol names, and geometries.
- *nets*—a list of all the nets in the design and the pins on the nets that provide continuity between components.
- *gates*—a record of all the schematic symbol instances on the schematic along with their logical/physical pin information. The *gates* design object contains swap information for every symbol instance in the design.
- *pkginst*—a copy of the gates information. *Pkginst* data can be used by LAYOUT for *Wasis* comparisons.
- *spares*—a list of spare and reserve parts in the design.
- *pkgs*—the statistics from the assignment process including the geometry and part number for each symbol, and the portion count. *Pkgs* data also includes the total number of each geometry type used in the packaging process, and the equivalent number of ICs.
- *notes*—a record of the notes assigned to schematic nets, pins, and instances. The *notes* design object is a good vehicle for the schematic designer to use in passing information to the layout designer. PACKAGE extracts note properties from the schematic and places the information in the *notes* design object.
- *geoms*—all the geometries used in the design.
- *catalogs*—a series of functions that tells PACKAGE to load certain catalog files. After the first PACKAGE design session, catalog files are automatically reloaded.

The *comps*, *gates*, *nets*, and *geoms* design objects are required input to the LAYOUT tool. The *gates* design object is optional.

Summary

In this module you learned about the PCB PACKAGE tool. We discussed the concepts of packaging. You learned about the *pkgconf* design object and how to use it as a filter for your input to the PACKAGE tool. We discussed both interactive and automatic packaging. You learned the steps involved in packaging your design. You also learned about the design objects that make up the output from a packaged design.

Lab Exercise

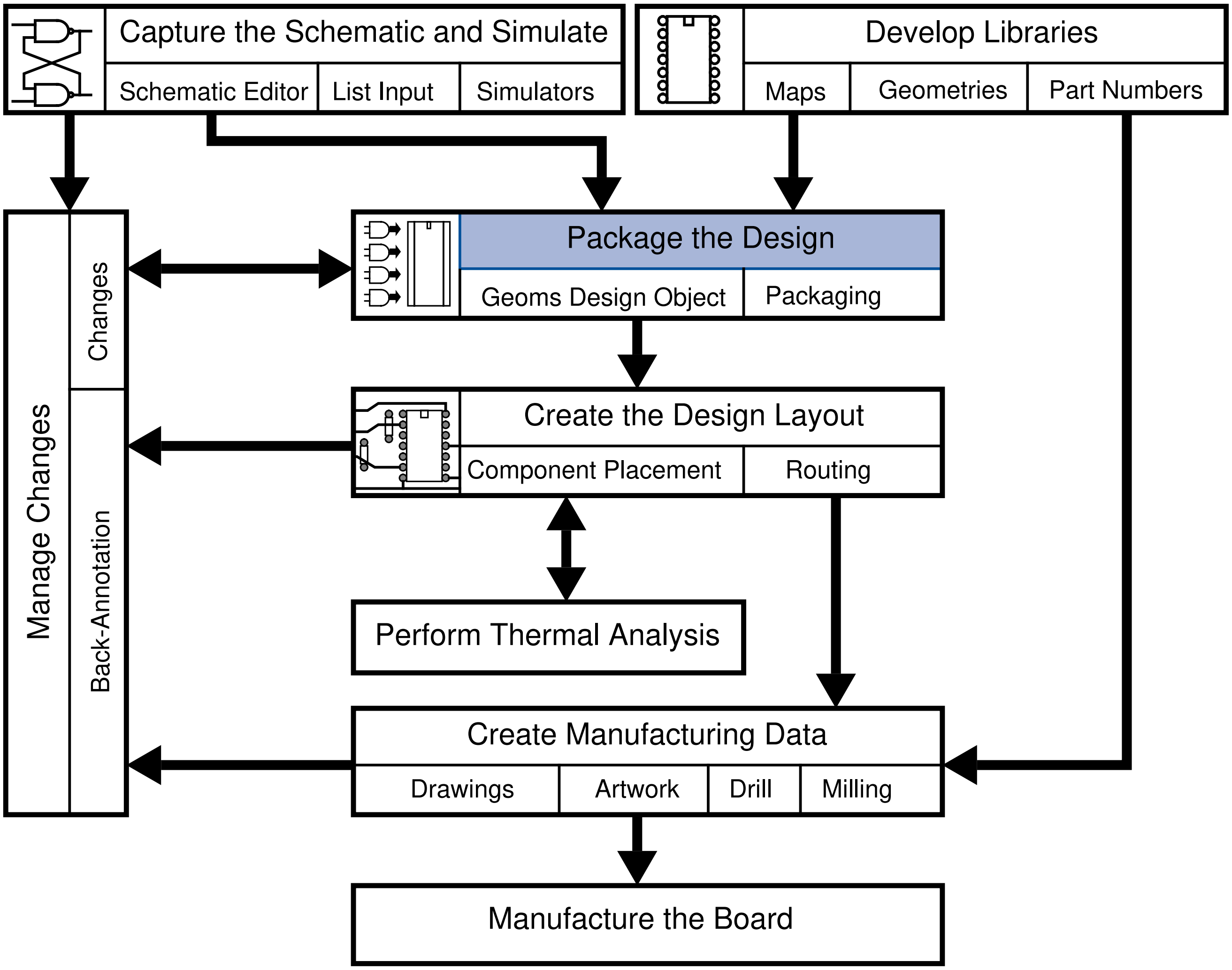
In this lab exercise you package the design, assigning all symbol instances to components. This process prepares a database for the board layout process. You also create a database of all geometries for the design by creating a *geoms* design object.

Upon completion of this lab exercise you can:

- Create a *pkgconf* design object in LIBRARIAN.
- Read required catalogs into PACKAGE.
- Build the design.
- Perform interactive packaging by changing reference designators.
- Add the Placement_region property to several gates.
- Perform Build again.
- Begin the process of creating a *geoms* design object by reading board and default vias into PACKAGE.
- Search the User_geom and Project_geom libraries to complete the *geoms* design object.

Turn to Module 4—Lab 1: "Packaging the Design".

Lab 1
Packaging the Design



Introduction

In this lab exercise, you package the design by assigning all symbol instances to components. This process prepares a database for the board layout process. You also create a database of all geometries for the design by creating a *geoms* design object, which is placed in the design directory.

Upon completion of this lab exercise you can:

- Create a *pkgconf* design object in LIBRARIAN.
- Read required catalogs into PACKAGE.
- Build the design.
- Perform interactive packaging by changing reference designators.
- Add the Placement_region property to several gates.
- Perform Build again.
- Begin the process of creating a *geoms* design object by reading board and default vias into PACKAGE.
- Search the User_geom and Project_geom libraries to complete the *geoms* design object.

Procedure

In this procedure, you invoke LIBRARIAN and create the *pkgconf* design object.

Preparation for Lab

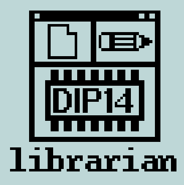
To set up for this lab you need to invoke the LIBRARIAN tool on your design. You then create the *pkgconf* design object.

1. If you or your instructor have not already done so, complete the Installation Procedure in the About This Training section of this manual.

2. Invoke the Design Manager by entering the following in a shell:

   ```
   Sys V> $MGC_HOME/bin/dmgr
   ```

3. Using the Design Manager, change your current directory to the board_new directory by clicking on the four-way icon in the navigator window. In the *Change directory to* dialog box, enter the pathname: your_path/**training/board_new/mod4** and **OK** the dialog box.



4. Find the LIBRARIAN icon in the Tools window and invoke it by placing the cursor on the icon and double clicking the Select (left, by default) mouse button.

5. In the Specify Invocation Mode dialog box that appears, choose **Invocation Mode: On a Design**. Press the **OK** button.

   The INVOKING LIBRARIAN: Select a Design dialog box displays.

6. Select the **sig_az** design and **OK** the dialog box.

   A Report-Startup message might appear in the middle of the LIBRARIAN Session window. This report is a list of notes concerning the files used to invoke the LIBRARIAN tool.

If there are errors in the report, you can ignore them. The errors occur because the design does not yet have some of the geometries saved as part of the geoms design object. The geoms design object can be saved in LIBRARIAN, but you will read additional geometries into the design in this lab exercise.

7. After reading the report notes, close the report window, and then maximize the size of the LIBRARIAN session window to fill the display.

Creating the Package Configuration File

The *pkgconf* design object must be created before running PACKAGE. If you invoke PACKAGE without the *pkgconf* design object in the design directory, PACKAGE looks for a system default file call *pkg.config*. This default file is located in the directory *$MGC_HOME/pkgs/pcb_base/data/pkg.config*. The default *pkg.config* contains no keywords, so all defaults are in effect.

You can customize the system default by editing *$MGC_HOME/pkgs/pcb_base/data/pkg.config*. You can also create a custom *pkgconf* design object for your design using the LIBRARIAN tool. This is what you do in the first part of the lab exercise.

You create or modify the *pkgconf* design object by completing a dialog box in LIBRARIAN.

1. Choose the **Setup Design Rules > Pkgconf...** menu item.

   The Setup Package Configuration dialog box is displayed.

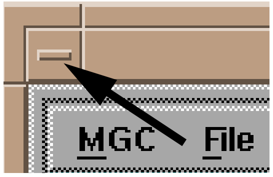
2. In the dialog box, specify the following, leaving all other options default, then **OK** the dialog box.

Table 1-2. Pkgconf Statements

Statement	Setting
Component Statement:	**No**
Net Statement	**No**
Connector Statement:	**No**
Annotate Portions:	**No**
Specify Units:	**No**
Ignore Props Statement:	**No**
Reference Statement:	**No**
Assign Prop Statement:	**No**
Alias Power Statement:	**No**
Instance Statement:	**No**
Pin Statement:	**No**
Equivalent Statement:	**No**
Write One Pin Nets to Nets Design Object:	**On**
Read Default Catalogs:	**On**
Use Properties to select Catalog part number:	**On**
Design Prop:	**On**
Dump Symbol Map:	**No**
Write all pins:	**Off**
Check mono symbols:	**Off**

3. Choose the **File > Save > Design Specify...** menu item. In the dialog box, select **Pkgconf** from the *Save to design* field, and then **OK** the dialog box.

   You have saved the *pkgconf* design object.



4. Close the LIBRARIAN session by choosing **Close** from the Session window menu. In the *Save Changes to Design?* dialog box, select **No**.

Starting PACKAGE

1. Invoke PACKAGE from the Design Manager tool window.

   A navigator dialog box displays.

2. Navigate to the name of the lab exercise design, **sig_az**. Select the design name **sig_az**, and **OK** the dialog box.

   An Optional Viewpoint Specification dialog box appears.

3. In the dialog box, use the default viewpoint and **OK** the dialog box.

   PACKAGE automatically creates a default PCB viewpoint.

If another dialog box requests you to **specify technology** *for the design you are packaging, choose* **Standard PCB** *and* **OK** *the dialog box.*

PACKAGE issues many messages and informational warnings as it invokes.

4. Read and **Close** the Notepad Report-Startup message.

Loading Catalogs

Create a link to access mapping and catalog information that exists in a separate catalog library.

1. Choose the **Catalogs > Add Catalog Link...** menu item. In the Add Catalog Link dialog box, enter the following:

   Catalog Name: **trng**

   Package assumes a *.catalog* extension.

   Pathname to Existing Library:

Enter only the pathname of the directory containing the catalog; do not include the catalog name here. It might be helpful to use the navigator, rather than typing the pathname.

`your_path`**/training/board_new/mod4/sig_az/pcb_parts/user_maps/trng_maps**

   Add to: **User**

   Directory type: **Permanent**

2. **OK** the dialog box.

3. Choose the **Setup > Catalog Library > Load Catalog Library...** menu item. In the dialog box, choose the following, and then **OK** the dialog box.

   All Catalogs From Directory

   User

   Design

   Check All Part Numbers Found

   Overwrite Duplicate Part Numbers

4. Look for errors and warnings in the Report-Saved message window. Also verify any notes, then **Close** the Notepad window.

   There are no errors, although there might be some warnings. All parts are found.

Reference Designator Text Setup

Before you perform automatic packaging, set up some reference designator prefixes for some symbols. First set up prefix text for capacitors.

1. Choose the **Setup > Reference Text** menu item.

   The Setup Reference Designator Text dialog box appears.

2. Select the **Set Default Reference Text** button. Enter the following:

   Default Reference Designator of **C**
   for Symbol **CAP?**
   for Symbol **PCAP?**

3. **OK** the dialog box.

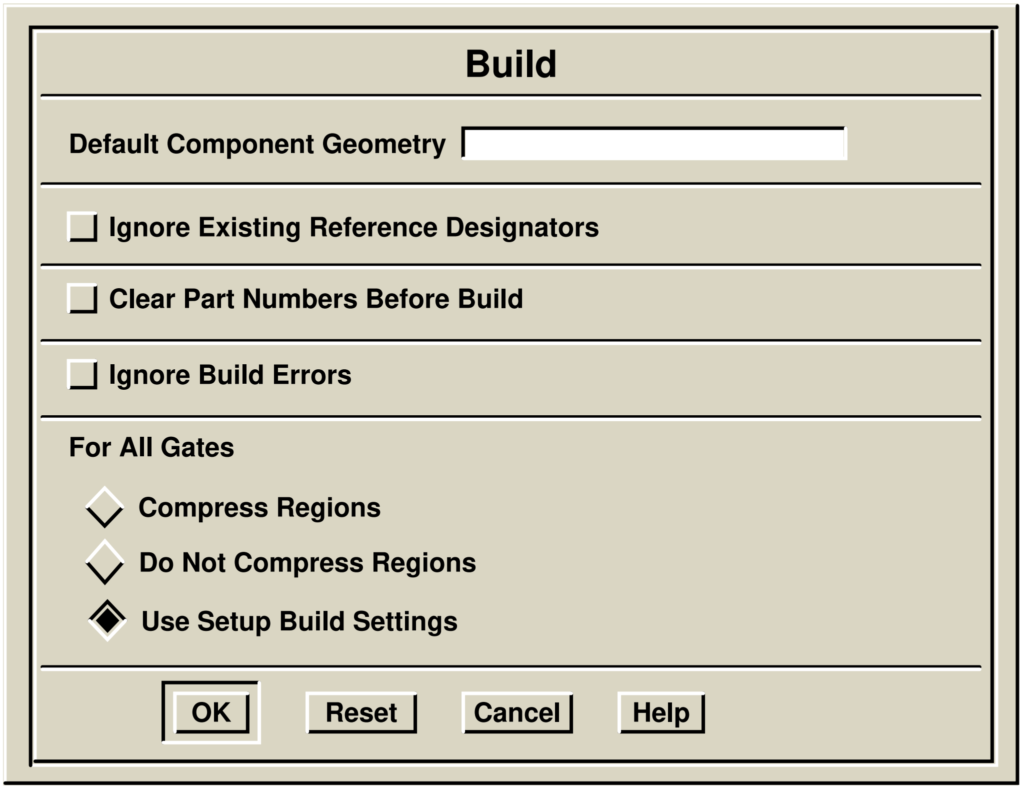
4. Choose the **Setup > Reference Text** menu item again. Enter a Default Reference Designator of **R** for Symbol **RES?**.

5. **OK** the dialog box.

Automatic Packaging

Whenever you enter PACKAGE on a design and all components have associated part numbers from the loaded catalogs, you need to run the automatic Build. The Build command packages the gate information into components. Any time you make a change to that packaging, you must perform another Build to verify the changes.

1. Choose the **[Component Summary] Build...** popup menu item. In the dialog box use only the default settings as shown in Figure **Figure 1-15.** Build Dialog Box, and **OK** the dialog box.



Build

Default Component Geometry

□ Ignore Existing Reference Designators

□ Clear Part Numbers Before Build

□ Ignore Build Errors

For All Gates

○ Compress Regions

○ Do Not Compress Regions

● Use Setup Build Settings

Figure 1-15. Build Dialog Box

Build works as an iterative process.

A report message window displays. You might see many warnings about physical pins on certain gates not being in the map, and about physical pins being reassigned. The build automatically resolves each of these, and you see that at the end of the build all the pins and mapping files are resolved with no errors.

2. After you have read the report and verified any notes, **Close** the Notepad report window.

   Each time you perform a Build, a report is generated.

   The Package Component Summary window now shows geometry, part number, and other information for the components.

Interactive Packaging

During the packaging process, you have the ability to make changes to the components. You can change the part numbers, geometries, and references at the gate level. In the first example, you change the geometry used to package a symbol type.

When PACKAGE invokes, the Component Summary window displays. This window shows all symbol types used in the design. When you select a symbol, the window changes to the Edit Symbol window. The Edit Symbol window shows the individual gates for the selected symbol.

Changing a Geometry

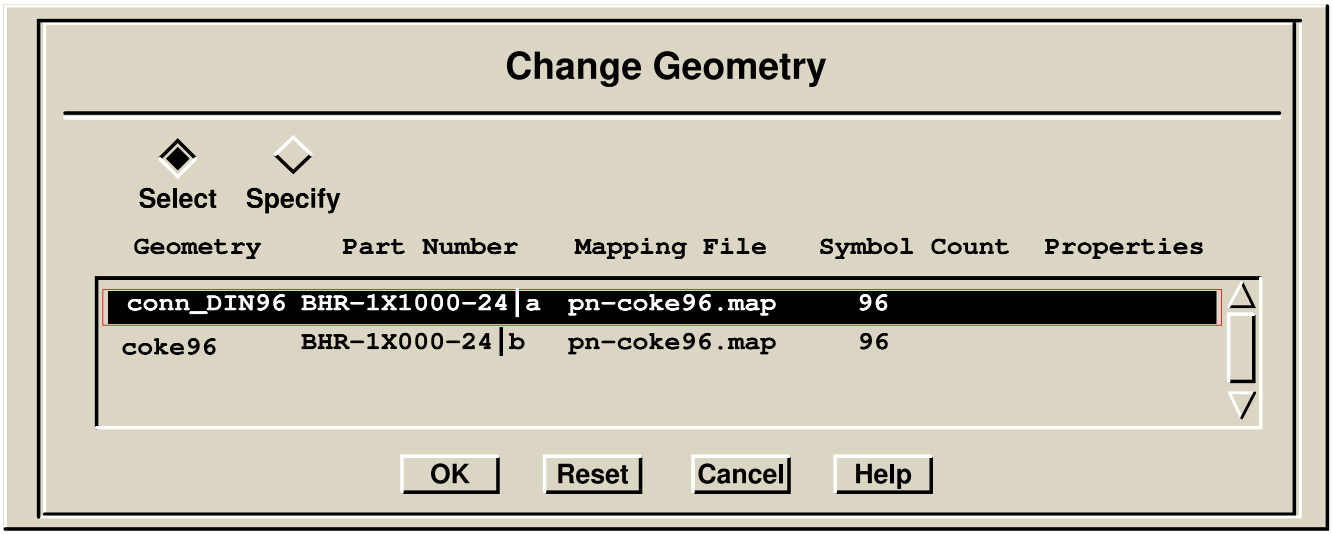
1. Place the cursor on the line in the Component Summary window that defines the connector symbol **CONN**, and click the Select mouse button.

   The Package Component Summary window changes to the Edit Symbol window for the Conn symbol. You can see that each symbol uses the coke96 geometry. You are going to change the geometry for all instances of the symbol.

2. Choose the **[Edit Symbol] Select > All Lines** menu item.

   The lines highlight to show the selected instances.

3. Choose the **[Edit Symbol] Change > Geometry...** menu item. In the Change Geometry dialog box, select the **conn_DIN96** geometry.



◆ Select ◇ Specify

```
Geometry      Part Number      Mapping File    Symbol Count   Properties
conn_DIN96 BHR-1X1000-24|a  pn-coke96.map      96
coke96     BHR-1X000-24|b   pn-coke96.map      96
```

Figure 1-16. Selecting an Alternate Geometry

4. **OK** the dialog box.

   The geometries change to conn_DIN96 for all symbol instances. You could have selected only one instance, and changed the geometry for that instance. Close the symbol, because you are going to modify another symbol later.

5. Choose the **[Edit Symbol] Close Symbol** menu item.

   In the Component Summary window, the symbol CONN is unassigned. You do a Build later to reassign the symbols.

Changing a Part Number

1. Select the symbol **PAL16L8A**.

   The Edit Symbol window for the PAL16L8A symbol displays.

2. Select all the symbol instances by dragging the cursor across all the lines in the Edit Symbol window.

3. Choose the **[Edit Symbol] Change > Part Number...** menu item. In the dialog box, select the **BHR-8X4600-29|b** part number, and then **OK** the dialog box.

4. Choose the **[Edit Symbol] Close Symbol** menu item.

Modifying Reference Designators

Next you modify reference designations.

Note the current range of reference designator text.

1. In the Package Component Summary window, select the **PCAP** symbol to open the Edit Symbol window.

   Notice that the reference designators are currently C96 through C99. You are going to change them.

2. Choose the **[Edit Symbol] Select > All Lines** menu item.

3. Choose the **[Edit Symbol] Change > Reference By Range...** menu item. Fill in the dialog box as follows, and then **OK** the dialog box.

   Leading Text: **C**
   Trailing Text: **[Leave blank]**
   Starting Number: **100**

   The reference designators are now C100 through C103.

Note the new range of reference designator text.

4. Choose the **[Edit Symbol] Close Symbol** menu item to return to the Component Summary window.

Resolving the Changes

Now you need to resolve all the changes you have made by doing a Build. You can wait until all the changes are done, and do one Build, or you can do Builds after each change.

1. Choose the **[Component Summary] Build...** menu item, then **OK** the dialog box using the default settings in the Build dialog box.

   There are no errors, but there are some notes.

2. Close the report window.

Adding Properties

In addition to assigning properties on the schematic using Design Architect, you can add the properties in PACKAGE or in LAYOUT. The following example shows how to add properties in PACKAGE.

Properties help control part number selection.

Properties restrict part number selection for a symbol to those part numbers in the catalog file containing matching properties and values as those on the symbol. To restrict part number selection, add the Value property to symbol Res_sip_pullup.

1. Select the **Res_sip_pullup** symbol to open it.
2. Choose the **[Edit Symbol] Select > All Lines** menu item.

   Now you add the Value property.

3. Choose the **Properties > Add/Change Property on Gate...** menu item. In the Add/Change Property to Gate dialog box, enter the following, and then **OK** the dialog box.

   Specify Name

   Name: **Value**

   Value: **330**

   To see the added properties, scroll the Edit Symbol window to the right.

4. Close the symbol.
5. Build the design again to complete the process.
6. Read the Notepad report, then close the report window.

Resolving Geometries

After packaging the gates, you resolve geometries and write the *geoms* design object. All geometries needed for the design must exist in the *geoms* design object before you enter LAYOUT. You can resolve geometries in PACKAGE or in LIBRARIAN. Resolve Geometries reads in the geometries defined by the catalog entries of packaged components. The menu item and function to resolve the geometries is the same in both PACKAGE and LIBRARIAN.

Creating Links to Additional Geometry Libraries

Create links to three libraries in the *user_geom* directory. These are geometry libraries into which you saved geometries using the LIBRARIAN tool.

Linking to Other Libraries

1. Choose the **Geometries > Add Other Library Link...** menu item. In the dialog box, use the navigator to enter a pathname to:

   ```
   your_path/training/board_new/mod4/sig_az/
   pcb_parts/user_geom/components
   ```

2. Now create links to two additional geometry libraries under *user_geom*: **padstacks** and **trng.**

Resolve the Geometries

1. Choose the **Geometries > Resolve Geometries...** menu item. In the Resolve Geometries dialog box, choose **Default**, and then **OK** the dialog box.

   If you save your geometries in libraries in the standard library hierarchy, you use the default option; otherwise you use the specify option. In this lab, use the default.

   This menu item parses all standard geometry directories, starting with the User directory, to find the required components and padstacks.

   The system records the geometries read and reports problems that occur while resolving the components and padstacks.

2. Read the Report-Saved messages, then **Close** the Notepad window.

   The report first states that some geometries were not found. PACKAGE attempts to resolve them and finds the geometries. PACKAGE then checks interdependencies again, and states that one fewer geometries were not found. Once again PACKAGE attempts to resolve them, and this time it finds all required geometries. This process of resolving and locating geometries is iterative.

Reading in Additional Geometries

Even when PACKAGE reports that all necessary geometries are found, it does not automatically read in vias, the board geometry, or the artwork order. In this section of the lab, you read in a via and the board geometry. You create the artwork order later using the FabLink tool. It is not necessary to have an artwork order at this point in the design process.

1. Choose the **Geometries > List Geometry Libraries...** menu item.

   The Default Geometry Library Hierarchy dialog box displays.

2. Click the Select mouse button to select **Other Geometry Libraries**.

   The Other Geometry Libraries dialog box displays the *components padstacks,* and *trng* geometry libraries.

3. Click on the **padstacks** library.

4. Select **View**.

5. From the Geometries List, select the **via040015** geometry.

6. Select **Read** to read the via geometry into the PACKAGE session.

Did you get an error? If so, it means the via has already been read into the design. A via can be read automatically by the Resolve function if it is part of another geometry, such as a component geometry.

7. Now read in the board geometry **signal_analyzer** from the **trng** geometry library:

You are now ready to finish resolving the geometries.

Resolve the Additional Geometries

Next you run the Resolve Geometries menu item again to resolve any padstacks for the components that were just read.

1. Choose the **Geometries > Resolve Geometries...** menu item. In the dialog box, choose **Default**, and then **OK** the dialog box.

   A report window displays. At first it shows that a few geometries were not resolved. After an iterative process, all geometries are resolved.

2. Close the report window.

Viewing the Schematic from PACKAGE

When you save the results of your PACKAGE session at the end of this lab exercise, you back annotate the packaging information to the PCB design viewpoint. To see the changes take place in the viewpoint, open a schematic window within PACKAGE.

1. From the menu bar, select the **View > Schematic Sheet** menu item.

2. Fill in the prompt bar as follows:

   Hierarchy: **/**
   Sheet: **sheet2**
   Window Setup: **left_right**

3. **OK** the prompt bar.

The schematic window opens to the right of the PACKAGE session.

Opening a schematic window is not required to save data. We open a schematic window here simply to allow you to see the annotations update in the PCB Design Viewpoint.

Saving the Design

Now you save the packaging information in the design.

1. Choose the **File > Save > Design All...** menu item. In the dialog box that displays, choose **Yes** for **Back-annotate PCB Design Viewpoint**.

Back annotating the PCB design viewpoint with the new property information is only a benefit if you are using a schematic from Design Architect. You need a schematic on which to back annotate the properties.

2. **OK** the dialog box.

   A report window displays a listing of all that is saved. PACKAGE makes a final check of pin information before a design is saved.

3. Close the report window.

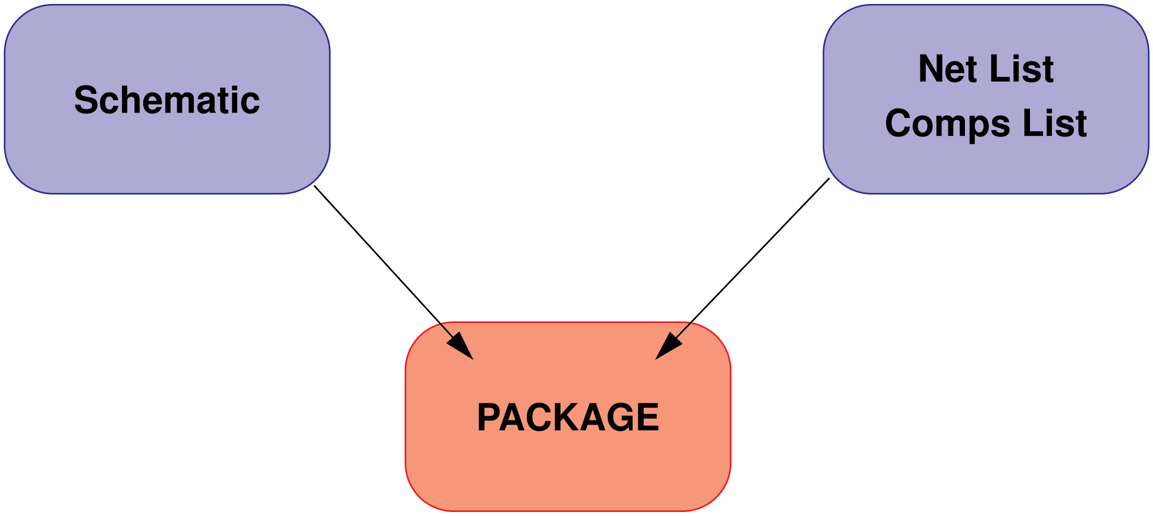
4. Close PACKAGE. You do not save changes again. Also close the Design Manager.

Congratulations! You have completed the *Packaging the Design* module. The next module of the *Board Station for New Users Training Series* is Module 5: *Managing Design Changes*.

Appendix A
Entering PACKAGE Without a Schematic

Appendix A discusses entering PACKAGE without a schematic. You use this method of design data entry only if you do not use output from a schematic created in Design Architect. If you use data only from Design Architect, this appendix is not necessary.

Appendix A teaches the expected format for the components and net list files and how to translate the files into design objects. It explains the concepts involved and provides a practical lab exercise.



Enter PACKAGE with either
a Net List and a Components List
or
a Design Architect schematic.

Objectives

Appendix A of the Packaging the Design for LAYOUT module examines design input from the *nets* and *comps* design objects. You translate a nets file and a components file from another schematic capture system into design objects the PACKAGE tool can use as input. This method provides an alternative to the Design Architect schematic for creating circuit data for the board design process.

In this appendix, you examine the format for both the nets file and the components file. You also learn to use the utility that takes these files and creates design objects in the *pcb* container of a design.

Requirements for Packaging Without a Schematic

If your design is not based on a Design Architect schematic and, therefore, has no design schematic, you can still invoke PACKAGE. However, you must have a components file and a nets file that PACKAGE can read in place of the design data.

The components file, the nets file, and the optional list of symbol instances, or gates, must reside in the *pcb* container under your design container as design objects named *comps*, *nets*, and *gates,* respectively. The *PCB Products Design Reference Manual* shows the formats for the *comps*, *nets*, and *gates* design objects.

Creating a Nets File

Table A-1. is a small portion of a nets file.

Table A-1. Nets File

```
# BOARD STATION NETS FILE FORMAT 1.0

NET '+15V' C97-1 C86-1 C69-3 U31-1 U70-7 C89-1 C83-2 C84-2 U30-8
NET '+15V' U29-11
NET '-15V' C96-2 C87-1 U69-4 U31-27 U70-4 C90-1 C99-2 C82-1 C85-1
NET '-15V' U29-14
NET '/AS' U68-59 U23-1 U23-19 U25-1 U25-19 U26-1 U26-19 U24-19
NET '/AS' U78-25 U80-25 U79-12
NET '/BR' U68-66 J2-2
NET ’/RW’ U55-11 U56-11 U57-11 U58-11 U59-11 U52-11 U53-11 U54-11
NET ’/RW’ U48-11 U51-11 U61-11 U44-11 U45-11 U46-11 U47-11 U40-11
```

A nets file consists of the following fields:

- **NET**—keyword at the beginning of each line. If the net is longer than one line, continue the net on the next line with the NET keyword and net name included.
- **net_name**—net_name identifies the net (wire). Names that do not start with the forward slash (/) character are global net names, like VCC and GROUND. Single quotes surround net names containing special characters.
- **reference-pin_number**—reference designation and pin name sets that are part of the net. The reference designation matches the component listing in the components file. Reference designations are separated from the pin_number by a dash (-) character. The pin_number is the physical pin number, which maps to the geometry.
- **properties**—the properties attached to the net (wire). Properties are listed at the end of the NET statement and are enclosed in parentheses. The property name is separated from the value by a comma, space, or both.

Creating a Components File

Table A-2. is a small portion of a components file.

Table A-2. Components File

Ref	Part Number	Symbol	Geometry
C1	BHR-1X0000-23	CAP	cc1210
C2	BHR-1X0000-23	CAP	cc1210
C3	BHR-1X0000-23	CAP	ck05
C96	BHR-1X0010-22	PCAP	cc0805
C97	BHR-1X0010-22	PCAP	cc0805
J1	BHR-1X1000-24	CONN	coke96
J2	BHR-1X1000-24	CONN	coke96
R1	BHR-1X2000-34	RESISTOR	rc1210
R2	BHR-1X2000-34	RESISTOR	rc1210
R7	BHR-1X4321-49-RC1210	RES_DIGITAL	rc1210
U3	BHR-1X4321-49-RZ090	RES.U	sip10

A components list consists of the following fields:

- **UNITS**—defines the database units of the components list. The following units may be specified: centimeters (CM), inches (IN), mils (ML), millimeters (MM), tmils (TM), and ten nanometers (TN).
- **Reference**—defines the reference designator for the component.
- **Part_number**—defines the part number used for the component. This number must match a part number in the catalog files loaded in PACKAGE.

- **Symbol**—equivalent to the Comp property value used on a schematic symbol. This value must match a Comp property value in the catalog files.
- **Geometry**—defines the component geometry created in LIBRARIAN or found in Mentor Graphics, company, or other libraries.
- **Board_location**—this optional field defines the component location (x,y), the placement side of board (1=top, 2=bottom), and a component rotation in degrees.
- **Properties**—the optional properties associated with the component. Properties are listed at the end of the line and are enclosed in parentheses. The property name is separated from the value by a comma, space, or both.

Converting Files to Design Objects

The *$MGC_HOME/bin/pcb_design_data_path* utility generates the *comps* design object from a components list, the *nets* design object from a net list, and the *gates* design object from an instance file.

To create the *comps* and *nets* design objects, the component list and the net list must reside at the same level as your design container. To create the *comps* design object, on the command line, enter:

```
$MGC_HOME/bin/pcb_design_data_path -objtype comps
   -writefrom <comp_file> -design <design_name>
```

To create the *nets* design object, on the command line, enter:

```
$MGC_HOME/bin/pcb_design_data_path -objtype nets
   -writefrom <nets_file> -design <design_name>
```

To create the *gates* design object, on the command line, enter:

```
$MGC_HOME/bin/pcb_design_data_path -objtype gates
   -writefrom <inst_file> -design <design_name>
```

PACKAGE automatically reads the *comps*, *nets*, and *gates* design object when it invokes if no design schematic exists in the design container.

Creating Gates

If your input to the PACKAGE tool is from a components file and a nets file, you need to create the information necessary for gate swapping. In the PACKAGE session, the Component Summary window lists the symbols in your design. Before you can open an Edit Symbol window on any of the symbols, you must create gates for the symbols. Two steps create the gates:

1. Load the catalog files needed for your design.

2. Create the gates for your design by choosing the menu item:

   [Component Summary window] > Create Gates

Once you create the gates, you can open an Edit Symbol window for any of the symbols in your design. In the Edit Symbol window, each individual gate has an instance identifier beginning with I. This I-number is the identifier Board Station uses to package each gate into a component.

Lab Exercise

In the lab exercise, you create the design objects that are the input to the PACKAGE process: the *nets* design object and the *comps* design object. The *nets* and *comps* design objects are usually created by net listing or design converter programs; however, the *nets* and *comps* design objects can be created with a text editor.

Upon completion of the lab exercise in Appendix B you can:

- View and edit a components file with the Notepad.
- View and edit a nets file with the Notepad.
- Add properties for board layout in both the nets file and components file.
- Create required design objects from the nets file and components file.

Procedure

In this procedure, you invoke an application and use the Notepad editor to create your design objects.

Preparation for Lab

To set up for this lab, you need to invoke an application so that you can use the Notepad editor. Because all applications have the Notepad editor, it does not matter what application you invoke. In this lab, however, you invoke the Design Manager. You can also create and edit the files using a UNIX text editor, as these files are ASCII format.

1. Invoke the Design Manager using the following shell command:

   `Sys V>` **$MGC_HOME/bin/dmgr**

2. When the Design Manager is ready, navigate in the navigator window to your **training/board_new/mod4** directory.

The comp_file and the nets_file become design objects.

In addition to the design directory **sig_az**, you also see the components file, **comps_file**, and the nets file, **nets_file**. To create the *comps* and *nets* design objects that you need in the design directory, the components file and nets file must reside at the same level as your design directory. For this lab, you have been supplied with these two files.

Editing a Components File

Now you open the existing components file using the Notepad editor.

1. Choose the **MGC > Notepad > Open > Edit...** menu item.

   The Select file to edit dialog navigator is displayed.

2. From the *Select file to edit* dialog navigator, select the **comps_file** and **OK** the dialog box.

   The *comps_file* is opened, and its contents are displayed in the notepad window.

   Now you save this file under a new name, and edit the new file. The *comps_file* remains in its original condition, unedited, so that you have a backup.

3. Choose the **File > Save As** menu item. In the *Save document as* dialog box that is displayed, enter the file name **components**, and then **OK** the dialog box.

   The pathname at the top of the notepad window changes to show the new name, *components*. The original file, *comps_file*, is not changed. You made a copy of it. Now you edit the contents of the *components* file.

4. Study the format of the file.

   A portion of the contents are also shown in **Table A-3.** A components file can be generated by another system or you can create one.

Table A-3. Comps File Format

Ref	Part Number	Symbol	Geometry
C1	BHR-1X0000-23	CAP	cc1210
C2	BHR-1X0000-23	CAP	cc1210
C3	BHR-1X0000-23	CAP	ck05
C96	BHR-1X0010-22	PCAP	cc0805
C97	BHR-1X0010-22	PCAP	cc0805
J1	BHR-1X1000-24	CONN	coke96
J2	BHR-1X1000-24	CONN	coke96
R1	BHR-1X2000-34	RESISTOR	rc1210
R2	BHR-1X2000-34	RESISTOR	rc1210
R7	BHR-1X4321-49-RC1210	RES_DIGITAL	rc1210
U3	BHR-1X4321-49-RZ090	RES.U	sip10

You might want to add additional data to this file. For example, you might want to add some properties to aid in component placement, such as the Placement_region property.

5. Edit the components file to add the Placement_region property to the components listed in Table A-3., using the following format:

```
R1 BHR-1X2000-34 resistor rc1210 ($G, "$V")
(placement_region, "ANALOG")  (INSTPAR, "23.7K")
```

Copy using the Edit>Copy to Cursor menu item.

When using the Notepad, instead of retyping each property, you might find it is easier to type the property and its value once, select it using the Select mouse button, move the cursor to where you want a copy and click the Select mouse button there, and then choose the **Edit > Copy to Cursor** menu item. You might also want to practice with the other copy and paste features. These menu items also exist in the Notepad's popup menu.

6. When you finish editing the file, choose the **File > Save** menu item to save the components file. Close the components file.

Editing a Nets File

Now you are going to edit the nets file using the same procedure you used to edit the components file. First you open the existing file. Then you save it under a new name, and edit the new file. Finally, you save the edits.

1. Choose the **MGC > Notepad > Open > Edit...** menu item.

   The Select file to edit dialog navigator displays.

2. From the *Select file to edit* dialog navigator, select the **nets_file** and **OK** the dialog box.

   The *nets_file* opens and its contents display in the notepad window.

   Now you save this file under a new name, and edit the new file. The *nets_file* remains in its original condition, unedited, so that you have a backup.

3. Choose the **File > Save As** menu item. In the *Save document as* dialog box that is displayed, enter the file name **net_list**, and then **OK** the dialog box.

   The pathname at the top of the notepad window changes to show the new name, net_list. The original file, nets_file, is not changed. You made a copy of it. Now you edit the contents of the file, net_list.

4. Study the format of the file.

   A portion of the contents are also shown in **Table A-4.** A nets file can be generated by another system or you can create one.

Table A-4. Nets File Format

```
#  BOARD STATION NETS FILE FORMAT 1.0

NET '+15V' C97-1 C86-1 C69-3 U31-1 U70-7 C89-1 C83-2 C84-2 U30-8
NET '+15V' U29-11
NET '-15V' C96-2 C87-1 U69-4 U31-27 U70-4 C90-1 C99-2 C82-1 C85-1
NET '-15V' U29-14
NET '/AS' U68-59 U23-1 U23-19 U25-1 U25-19 U26-1 U26-19 U24-19
NET '/AS' U78-25 U80-25 U79-12
NET '/BR' U68-66 J2-2
NET ’/RW’ U55-11 U56-11 U57-11 U58-11 U59-11 U52-11 U53-11 U54-11
NET ’/RW’ U48-11 U51-11 U61-11 U44-11 U45-11 U46-11 U47-11 U40-11
.
```

You might want to add additional data to this file. An example is to add the Net_type property, **net_type**.

5. Add the Net_type property with the value **DIFF_PAIR** to the nets **/FRAME_DATA+** and **/FRAME_DATA-**, using the following format:

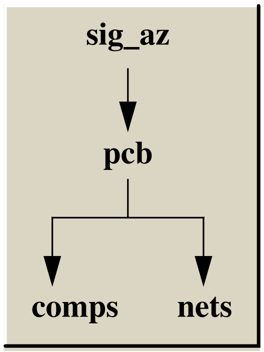
   ```
   NET '/FRAME_DATA+' J1-61 U1-5 (NET_TYPE, "DIFF_PAIR")
   NET '/FRAME_DATA-' J1-59 U1-4 (NET_TYPE, "DIFF_PAIR")
   ```

   For additional information on properties, refer to module *Placing Components on a Circuit Board* module in the *Board Station for New Users Training Workbook*, or section "Properties" in the *PCB Products Glossary and Index*. The "Properties" section of the *PCB Products Glossary and Index* is constructed like an index. Under the Properties heading (the section name), you find entries directing you to other PCB documentation, depending on the type of property information you need.

6. Choose the **File > Save** menu item.

7. Close the Notepad window and the Design Manager.

Creating the Design Objects



You have already created a design directory earlier in LIBRARIAN, called *sig_az*. If your design schematic came from Design Architect, this directory is created for you. This directory contains all the layout information for your design. If your design is not based on a Design Architect schematic, it does not have a design file, *pcb_design.vpt*, to use for packaging the design. Your design is also missing the *comps* and *nets* design objects.

Because you are not using a schematic from Design Architect, you created the components file and nets file. Now you need to use those two ASCII files to create the design objects *comps* and *nets* and place those design objects in the *pcb* container. The *pcb* container is the directory named *pcb* in the *sig_az* design directory.

Because your design is not coming from Design Architect, you need to provide design objects that would otherwise be created for you.

The three design objects are:

- *comps*: You create this from the components ASCII file you made earlier.
- *nets*: You create this from the components ASCII file you made earlier.
- *pcb_design.vpt*: If this does not exist, PACKAGE creates it for you when you invoke the application.

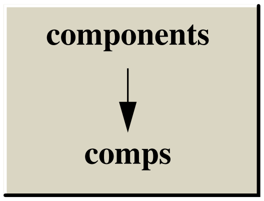
1. Change your current directory to the *mod4* directory by entering the following shell command in a UNIX shell:

   cd your_path/training/board_new/mod4

2. List the contents of the *mod4* directory by entering the following shell command:

   ls

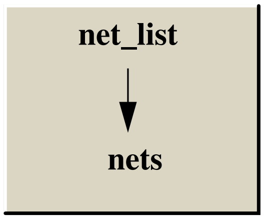
   You can see the *components* and *net_list* in addition to the *sig_az* directory.



3. Create the *comps* design object by entering the following shell command:

```
$MGC_HOME/bin/pcb_design_data_path  -objtype comps
-writefrom components  -design sig_az
```

   You see a message stating that the requested design object is updated, and the message lists the name of the new file.



4. Create the *nets* design object by entering the following shell command:

```
$MGC_HOME/bin/pcb_design_data_path  -objtype nets
-writefrom net_list  -design sig_az
```

   During the next lab, you package the design using the *comps* and *nets* design objects you just created.

Summary

This appendix covered the techniques for creating a design input for the circuit board design process from a nets file and a components file. These files, when converted to *nets* and a *comps* design objects, are the typical inputs from other design systems. The Board Station tools accept either a Design Architect schematic or a nets file and a components file as design input.

INDEX

欢 迎 光 临

微波 EDA
http://www.mweda.com

EDA学习网
http://edastudy.ik8.com

http://www.mweda.com　　http://edastudy.ik8.com

本站专业提供各种微波仿真软件和 PCB 设计软件的安装光盘和学习培训教程，本站提供的所有教程教材都是公司培训用的，很系统的讲述了相关软件的应用。主要项目有：

破解软件类：微波仿真软件

ADS2004A　ADS2003C　ADS2003A
HFSS9.2　HFSS9.1　HFss9.0　HFSS8.0
Ansoft Designer1.1　Ansoft Serenade8.71　Ansoft Maxwell 10　Ansoft SIWave
Microwave Office2002　Sonnet Suite Pro 9.52　CST5.0
Super NEC 2.5　Zeland IE3D9.2　XFDTD 6.0

破解软件类：PCB 工具软件

Mentor EN2004　Mentor EN2002　Mentor ePD2004　Mentor SDD2004
Mentor WG2004　Mentor WG2002　Mentor ISD2004
PowerPCB5.0　PowerLogic5.0　Orcad10.3
PADS2005　PADS2004
Cadence SPB15.2 (Allegro 15.2)　Cadence PSD15.0 (Allegro 15.0)

软件学习、培训教程：

Mentor EN:　Mentor EN 原版培训教程
Mentor EN 视频教程

Mentor WG:　Mentor Expedition / Mentor WG 中文用户手册
Mentor Expedition (WG) PCB Training Workbook
Mentor DxDesigner Design Processing Training Workbook

HFSS：　HFSS 9.2 入门与提高教程
HFSS 9.0 入门与提高教程
HFSS9 视频教程
HFSS9 Training Tutorials
HFSS8 Training Manual
电子科大 HFSS8 教程

ADS:　ADS 中文基础教程
ADS 设计实验教程
ADS 入门教程
ADS2003 Training　Workbook: ADS2003 fundamental
ADS　Training Workbook　for Momentum
ADS Customization Training Workbook

Using ADS Communication Systems Designer

Using ADS to Design WCDMA/3GPP Communication Systems

Ansoft Designer:

Ansoft Designer 2.1 Full Book

Ansoft Designer System Training Guide

Ansoft Designer 1.1 入门与提高

Ansoft Designer 多媒体教程

Ansoft Designer Training 讲义

Cadence Allegro:

Allegro 15.2 原版教程

Concept-HDL 15.2 原版教程

Allegro 14.2 原版教程

Allegro 视频教程

Allegro PCB Layout 高速电路板设计

注： 本站还有Ansoft Maxwell, Ansoft Serenade, Ansoft Q3D，Zeland IE3D, PowerPCB，Orcad等的培训教程。以及射频/微波/天线类英文原版书籍。详情可登陆：http://www.mweda.com 或 http://edastudy.ik8.com